UNIVERSITÀ
DEGLI STUDI
FIRENZE

Università degli Studi di Firenze
Dipartimento di Ingegneria dell'Informazione (DINFO)
Corso di Dottorato in Ingegneria dell'Informazione

Curriculum: Telecomunicazioni
SSD: ING-INF/03

Video forensic tools exploiting features from video-container to video-encoder level

Candidate
Dasara Shullani

Supervisor
Prof. Alessandro Piva

PhD Coordinator
Prof. Luigi Chisci

---

ciclo XXX, 2014-2017

Università degli Studi di Firenze, Dipartimento di Ingegneria
dell'Informazione (DINFO).

Hard Work
High Expectations

Abstract

The escalation of multimedia contents exchange, especially of videos belonging to mobile devices, and the availability of a great amount of editing software has raised grave doubts on their digital life-cycle. In this thesis, we firstly introduce a new dataset for multimedia forensics and then develop forensic tools that analyse the video-container and the video-signal in order to evaluate possible tampering that have been introduced in the life-cycle of a video content.

The first contribution consists on the release of a new Dataset of videos and images captured from to 35 modern smartphones/tablets belonging to 11 different brands: *Apple*, *Asus*, *Huawei*, *Lenovo*, *LGelectronics*, *Microsoft*, *OnePlus*, *Samsung*, *Sony*, *Wiko* and *Xiaomi*. Overall, we collected 11732 native images; 7565 of them were shared through Facebook, in both high and low quality, and through WhatsApp, resulting in a total of 34427 images. Furthermore we acquired 648 native videos, 622 of which were shared through YouTube at the maximum available resolution, and 644 through WhatsApp, resulting in a total of 1914 videos. The uniqueness of the VISION dataset was tested on well known forensic tool, i.e., the detection of the Sensor Pattern Noise (SPN) left by the acquisition device for the source identification of native/social media contents.

The second contribution is based on the analysis of the container structure of videos acquired by means of mobile devices. We argue that the atoms belonging to the container, in terms of order and value, are fragile and that it is more difficult to hide their modifications than the regular metadata. This characteristic can be exploited to perform Source Identification and Integrity Verification of videos taken from devices belonging to well known operating systems and manufactures.

In the third contribution we focus on the video-signal and on its encoding process. We used codecs that perform a hybrid video coding scheme, and developed a classification technique able to perform group of picture length estimation and double compression detection. The proposed technique is one of the fastest approaches that use videos encoded with B-frames, with both constant bit rate and variable bit rate.

Contents

Chapter 1

Introduction

In the last decades, the huge technological advancements in the communication and information fields have allowed the burst of digital contents, such as images and videos to the point of making them to become the preferred means to share information. Given their digital nature, these data also convey several information related to their life cycle such as the source device or processing they have been subjected to, which are studied by multimedia forensic algorithms.

1.1 The objective

Multimedia Forensics (MF) is a research area aiming to identify traces, left by the digital history of an image or video, by exploiting well known tools designed for digital contents analysis and multimedia security. In particular, this thesis is focused on video forensics, that still has to face several uncharted matters with respect to the image forensics, mainly due to the wider range of possible alterations that can be performed on such digital contents, and to the higher types of encoding standards. As this discipline grows, different kinds of analysis have become possible; in the following we list those that gathered more interest and that will be discussed in the thesis:

Source Identification : its objective is to determine from which device and/or which brand/model of device a specific multimedia content has been acquired with.

Source Classification : it aims to classify multimedia contents according

to some characteristics of the originating source, such as the operating system (e.g. Android versus *Apple*), camera model (e.g. *Huawei P9* versus *HuaweiP8*), etc.

Integrity Verification : its objective is to identify the presence of possible alterations (malignant or not) that a multimedia content has undergone, such as double compression or other post processing procedures, which distinguishes these contents with respect to the pristine one.

1.2 Contributions

This thesis proposes three main contributions to the aforementioned video forensic issues:

- *VISION dataset*, a new image and video dataset useful for benchmarking multimedia forensic tools will be presented. We have collected and made available to the research community a large dataset composed by thousands of images and videos acquired from portable devices of most famous brands, including those featuring in-camera digital stabilization. We also provide the *social version* of most contents, obtained by uploading and downloading them to/from well known social media platforms, namely Facebook, YouTube and WhatsApp. In addition, we will discuss the results of some state-of-the-art forensic applications such as Image and Video Source Identification obtained by using this dataset, that will highlight some interesting issues that have not been noted until now, since there were no sizeable benchmarks available in the research community.

- *Video forensics based on Low-Level features.* Low-level features, such as the container structure and content, can be extracted from a digital video and will be exploited as a mean to video forensics problems. In particular, these features will be used to determine *the integrity* of video content acquired with a mobile device and subsequently uploaded and downloaded through social-platforms. Furthermore, these low-level characteristics will be exploited to distinguish the Brand/-Model acquisition source, thus achieving Brand/Model identification and Brand/Model classification.

- *Video forensics based on High-level features.* High-level features are derived from the encoding algorithm prediction types; The Variation of Prediction Footprint [65] will be extended designing an algorithm based on Support Vector Machine classification that will allow to evaluate the footprint introduced by the double compression on B-frames, thus solving issues of integrity verification also on video using bidirectional prediction.

Chapter 2

Literature review

In this Chapter we will give a brief introduction on the main concepts of video encoding and video container structure, then an overview on the state of the art related to Video Forensics will be provided, discussing solutions proposed by the research community for what concerns problems of video source identification and classification, and video integrity detection.

2.1 Background on digital videos

A digital video is essentially a set of still images, called frames, displayed in rapid sequence. The reproduction speed must be high enough to exploit the *vision persistence* phenomenon, in this way the human brain is able to assemble images to perceive the fluid movement of a video content. Since the storage of a video as a sequence of still image is too onerous, a set of compression algorithms, such as MPEG-2 [41], MPEG-4 [40], H.264 [42] and H.265 [43], has been developed. A compression algorithm, also called codec, is composed of an encoder function that converts the source data into a compressed form occupying a reduced number of bits, prior to transmission or storage, and a decoder function that converts the compressed form back into a representation of the original data.

The video can be exchanged in its native form, as an H.264 stream for example, or by means of a *container*, such as MP4 [3] or MOV [10], that concatenates video metadata information, audio streams, subtitle streams and even multiple video streams in a single multimedia file.

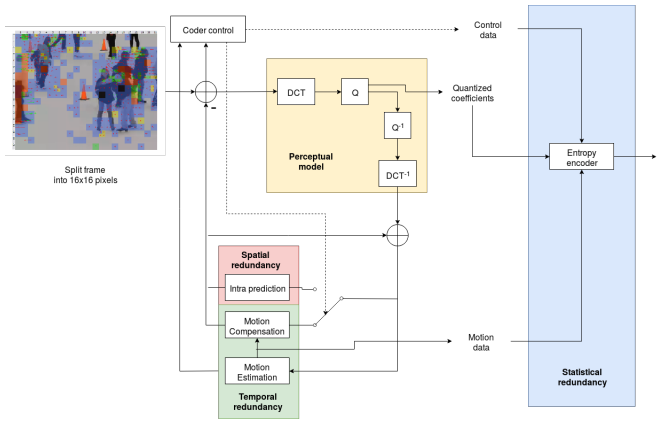


Figure 2.1: An example of video enconding.

In the following we will describe the main aspects of the video encoding procedure and the container structure.

2.1.1 Video coding basics

A video content consists of frames of different type depending on their content; in particular we can distinguish between intra predicted frames (I-frames) and inter predicted frames (P-frames and B-frames). Each frame is analysed in non overlapping macroblocks (MBs) such as 16×16 pixels, according to a processing chain as the one depicted in Figure 2.1. A specific sequence of frames is called a GOP, group of pictures, typically an I-frame followed by a sequence of P and B frames. The total number of frames in a GOP is also called *GOP size*.

The algorithm of a video encoder consists of three main functional units: a prediction model, a spatial model and an entropy encoder, as depicted in Figure 2.1.

From an uncompressed video sequence, the prediction model attempts to reduce redundancy by exploiting the similarities between neighbouring video frames, typically by constructing a prediction of the current frame. The output of the prediction model is a residual frame, created by subtracting the prediction from the current frame, and a set of model parameters indicating the *intra prediction* type or describing *how motion* was compensated.

An I-frame, or intra predicted frame, is a single frame of digital content

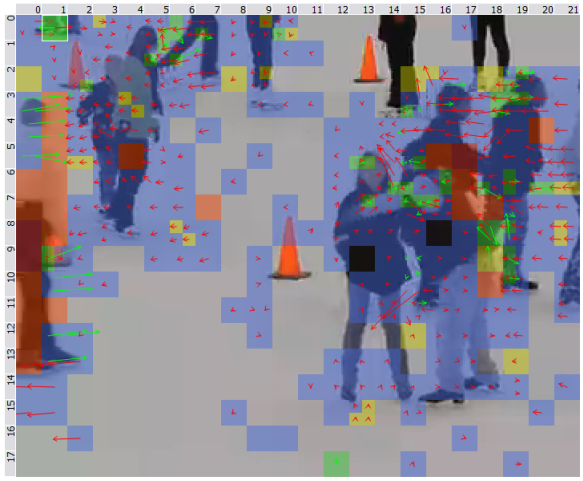


Figure 2.2: H.264 B-frame extracted from Ice.

that the encoder evaluates independently with respect to the neighbouring frames. A frame is called P, or predicted, if it contains *uni-predicted macroblocks*; namely the prediction of those blocks are carried out by a single reference MB. P-frames can be encoded with three types of macroblocks: I-MBs, or intra macroblocks, that are considered independent MBs; S-MBs that corresponds to skip macroblocks, also known to be the exact copy of some previous MB; P-MBs that are uni-predicted macroblocks referencing a similar macroblock in the past. For what concerns a B-frame, or bi-predicted frame, it contains MBs that are predicted by means of macroblocks located in the future, in the past or both. Similarly to the P-frames, in B-frames we can have the following type of macroblocks which are depicted in Figure 2.2[1]: I-MBs intra macroblocks in red; S-MBs skip macroblocks, not coloured; FW-MBs forward predicted macroblocks (also P-MBs), in blue with red arrows; BW-MBs backward predicted macroblocks, in blue with green arrows, that are predicted using a reference located in the future; Bdir-MBs bidirectional predicted macroblocks, in yellow, that combine two predictions, one from the past and one from the future.

[1] *Ice* raw video sequence downloaded from `https://media.xiph.org/video/derf/` on October 2017.

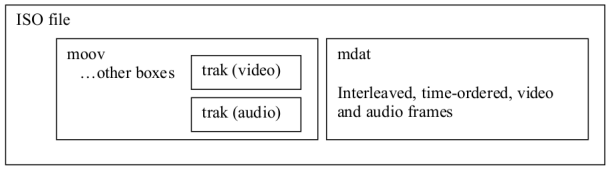


Figure 2.3: ISO Base Media File Format structure.

In the spatial model the residual frame is transformed and quantized, using for example the DCT - Discrete Cosine Transform, in order to reduce spatial redundancy. The transform converts the samples into another domain in which they are represented by transform coefficients. The coefficients are quantized to remove meaningless values, leaving a small number of significant coefficients that provide a more compact representation of the residual frame. Thus, the output of the spatial model is a set of quantized transform coefficients. The parameters of the prediction and spatial model, i.e. *intra prediction mode*(s)/ *inter prediction mode*(s), *motion vectors* and *quantized transformed coefficients*, are compressed by the entropy encoder. This removes statistical redundancy in the data, representing commonly occurring vectors and coefficients by short binary codes. As a result, the entropy encoder produces a *compressed bit stream* to be transmitted and/or stored.

On the other hand, the decoder reconstructs a video frame starting from the compressed bit stream. The coefficients and prediction parameters are decoded by an entropy decoder after which the spatial model is decoded to reconstruct a version of the residual frame. Then the prediction parameters, together with previously decoded image pixels, are used to create a prediction of the current frame and the frame itself is reconstructed by adding the residual frame to this prediction.

2.1.2 Video container

The MP4-like file formats, such as MP4, MOV and 3GP, are derived from the ISO Base Media File Format [4] (iso base), that is designed to contain time-based media contents, such as audio streams or video streams.

The *iso base* files can be characterized by three structures, namely a logical, a time and a physical structure. The logical structure of a video file consists of its time-parallel tracks; the time one is composed by the sample

Atom name and depth						Atom description description
Depth-1	Depth-2	Depth-3	Depth-4	Depth-5	Depth-6	
ftyp*						file type and compatibility
moov*						container for all the metadata
	mvhd*					movie header
	trak*					container for an individual trak
		tkhd*				track header
		mdia*				container for the media information
			mdhd*			media header
			hdlr*			handler
			minf*			media information container
				vmhd		video media header
				smhd		sound media header
				hmhd		hint media header
				dinf*		data information box
					dref*	data reference box
				stbl*		sample table box
					stsd*	sample descriptions
					stts*	time-to-sample
					stcs*	sample-to-chunk
					stcz	sample sizes (framing)
					stco*	chuck offset
					co64	64-bit chunk offset
					stss	sync sample table
					sdtp	independent and disposable samples
		udta				user-data
	udta					user-data
moof						movie fragment
mdat						media data container
free						free space
meta						metadata

Table 2.1: MP4-like Container structure.

sequences contained in each track, that are mapped to the overall video timeline by optional lists. The physical structure of the file separates the data needed for logical, time, and structural de-composition, from the media data (*mdat*) samples themselves. This structural information is concentrated in a movie box (*moov*), possibly extended in time by movie fragment boxes (*moof*). The movie box links the logical and timing relationships of the samples, and also contains pointers to where they are located.

A video file container describes its contents by means of a sequence of objects. All the data is contained in atoms, also called boxes, that are identified by a unique 4 bytes characters, as in Table 2.1, where we describe the meaning of each atom and the corresponding container depth.[2] In Figure 2.3 a possible MP4-like container structure at depth-1, is depicted; in particular:

ftyp atom: it is semi-mandatory and must be present and explicit as soon as possible in the file container because it defines the *best use* and compatibility of the content itself.

moov atom: it is the most complex atom of the container, It contains all the metadata needed for the decoding of the data stream.

mdat atom: it contains all elements of the data stream.

2.2 Multimedia Forensics applied to videos

In the following we will review the possible solutions given by the forensics community to the problems Source Identification, Source Classification and Integrity Verification. First of all, we will start in Subsection 2.2.1 with the description of the Datasets used for the *source identification problem*; in Subsection 2.2.2 we will take into consideration low level features belonging to video contents to solve both *source identification* and *classification*, in addition the *integrity verification*; finally in Subsection 2.2.3 we will have a look at solutions for the *integrity verification* that uses high-level video features.

[2] In Table 2.1 the asterisk (*) is used to identify atoms that are considered mandatory by the ISO, moreover an extended version can be found in [4].

2.2.1 MF Datasets

One of the first datasets adopted in the multimedia forensic community is the UCID database [56], originally designed for the evaluation of image retrieval techniques. Such dataset includes 1338 uncompressed images stored in the TIFF format, but their size is very small, either 512×384 or 384×512 pixels.

The first sufficiently large and publicly available image database specifically designed for forensic applications is the Dresden Image Database [31, 32]. This dataset includes images of various indoor and outdoor scenes acquired from 73 devices, selected from 25 camera models spanning most important manufacturers and quality ranges. All cameras were configured to the highest available JPEG quality setting and maximum available resolution, and, when supported by the device, also lossless compressed images were stored. The image resolution ranges from 3072×2304 to 4352×3264 pixels, for a total of 16961 JPEG images, 1491 RAW (unprocessed) images, 1491 RAW images processed in Lightroom 2.5 and 1491 RAW images processed in DCRaw 9.3. Since 2010, this dataset has been used by most of the works dealing with benchmarking of source identification methods.

More recently, RAISE (RAw ImageS datasEt) was presented [25]: it is a collection of 8156 raw images including a wide variety of both semantic contents and technical parameters. Three different devices (a Nikon D40, a Nikon D90, and a Nikon D7000) are employed, and the images are taken at very high resolution (3008×2000, 4288×2848 and 4928×3264 pixels) and saved in an uncompressed format (Compress Raw 12-bit and Lossless Compress Raw 14-bit) as natively provided by the employed cameras. Each image is also assigned one of seven possible categories, namely, "outdoor", "indoor", "landscape", "nature", "people", "objects" and "buildings". In the framework of the European project REWIND[3], a set of 200 uncompressed images acquired with a Nikon D60 camera were also made available [66, 67] (among other sets for splicing detection, copy-move forgeries and recapture videos, all including a few number of samples). There are also other datasets, not cited here, that have been designed more for image tampering detection than for source identification, and thus no or little information is provided about the device generating the images.

As to digital videos, in the literature there are very few datasets designed to be used in forensic scenarios; one of them is the SULFA [55], created by

[3] All related materials on the REWIND project can be found online at `https://sites.google.com/site/rewindpolimi/home`.

the University of Surrey. It collects 150 videos, each 10 seconds long, at 30 fps with a resolution of 320×240 pixels. The native videos are given compressed in H.264/AVC and MJPEG, for each camera: a Canon SX220, a Nikon S3000 and a Fujifilm S2800HD. Authors designed the dataset to be used for cloning detection, performed by means of Adobe Photoshop CS3 and Adobe After Effect CS5 [55]. The SULFA dataset was also extended by the REWIND dataset [14]; anyway these datasets are less interesting for video source identification, since they contain few digital cameras only and no smartphone, while we know smartphones are the most representative kind of device today, especially for applications on social media platforms. Recently the Video Tampering Dataset (VTD) was provided by Al-Sanjary et al. [8]. The VTD, focused on video tampering detection on videos collected from the YouTube platform, is composed by 33 downloaded videos, 16 seconds long, at 30 fps with a HD resolution. The original dataset is subdivided into four subsets: one containing unaltered videos; one with videos created by splicing; one with videos manipulated by copy-move, and one with videos tampered by swapping frames. Although they use a social media platform to acquire videos and provide interesting tampering techniques, there are not useful information related to the camera or device used.

The previous review shows that all currently available datasets consider mainly images, and the ones containing videos are not significant for video source identification; moreover, it is not possible to investigate relationships between images and videos acquired with the same sensor: this fact is a strong limitation, since 85% of shared media are captured using smartphones, which use the same sensor to capture both images and videos. Finally, another limit in the state-of-the-art is represented by the lack of a collection of controlled content coming from social media platforms, like Facebook, YouTube and WhatsApp; indeed, recent multimedia forensic applications would take advantage in having a large dataset containing such kind of contents: for instance, in [64] the authors address the performance of identifying the source of YouTube videos, but limiting to a scenario with videos belonging to 8 webcams of the same model (Logitech Quickcam STX). Similarly, Bertini et al. [12] propose to extract the Sensor Pattern Noise from images to identify fake social media accounts, but the technique was tested on 5 mobile devices only, with 200 images each.

2.2.2 Container features

When it comes to investigate the credibility of a digital video, two main categories of analysis exist: integrity verification and authenticity verification. The terms *integrity* and *authenticity* have been often treated like synonyms in the multimedia forensics research community, until recent time [48]; nevertheless, they have a strongly different meaning. According to the Best Practices for Image Content Authentication [61] of the Scientific Working Group on Digital Evidences[4], indeed, "content authentication is used to determine whether the visual content depicted in imagery is a true and accurate representation of subjects and events", while "integrity ensures that the information presented is complete and unaltered from the time of acquisition until its final disposition". There are cases where authenticity of a video is more important than its integrity (think to a video found on YouTube showing the preparation of a terrorist attack: the integrity is surely compromised by YouTube re-encoding process, yet the content could be authentic and worth of attention); there could be cases instead where integrity is more important than authenticity itself (for example, if previously mentioned video is found on a suspect's smartphone, it is of prominent importance to understand whether it is compatible, at least in terms of format and metadata, with the hypothesis of being captured by that smartphone or not). Noticeably, while it is possible to conduct authenticity analysis in a totally blind fashion (that is, without any background knowledge on the originating device), it is very difficult to reliably assess the integrity of a digital content without any information about its source: there is need for reference material to compare with, unless some obvious traces of file manipulation are present (e.g., the name of an editing software in the metadata).

Common techniques developed for the forensic analysis of digital videos mainly focus on the analysis of the data stream, i.e. the audio-visual signal, based on the detection of artifacts and inconsistencies in the (statistics of the) content. For example, identification of the source device is possible through sensor noise analysis [18], while detection of double encoding or manipulation can be done by analyzing prediction residuals [57] or macroblock types [29, 65].

[4]Formed in 1998, the Scientific Working Group on Digital Evidence brings together law enforcement, academic, and commercial organizations actively engaged in the field of digital forensics to develop cross-disciplinary guidelines and standards for the recovery, preservation, and examination of digital evidence.

A less explored approach is the analysis of the file format and metadata, to determine their compatibility, completeness, and consistency with respect to the context in which it is assumed the resource has been created. The techniques belonging to this class have been mainly borrowed from the analysis carried out on digital images. In particular, JPEG [2] and EXIF [27] metadata have been studied in several works: since each acquisition device and processing software usually adopts customized quantization tables, it is possible to exploit these differences to address the source origin problem [28]. By considering the number of EXIF entries as well as image and thumbnail compression parameters and sizes, *Kee et al.* [46] associate images whose origin is not known to a certain class of source devices. These studies have been limited by the fact that information contained in the metadata can be easily edited by a user in order to hide possible modifications, thus limiting the trustworthiness of such approaches.

Following studies revealed that also the file structures contain a lot of information about the history of a content, while being much more difficult to extract and modify from a user than metadata, since available editing software and metadata editors do not have a functionality to modify such a low-level information, like the internal order of the core file structures.

Indeed, both the acquisition phase and the subsequent post-processing steps modify the content and the structure of the file, such that these peculiarities can be exploited both for source identification and integrity verification. In particular, Gloe [30] demonstrated that the specific internal order within JPEG and EXIF structures represent a discriminating information for the authentication of a digital image.

These studies have been recently extended to digital videos too. In [33], *Gloe et al.* analyse the video files by exploring the low-level characteristics represented by metadata and low-level file format information such as the structure of the video file container. In particular, the authors noticed that the video standards prescribe only a limited number of characteristics for the data container formats, thus leaving a lot of freedom to the manufacturer. The authors, after analysing 19 digital camera models, 14 mobile phone models, and 6 video editing toolboxes, report considerable differences in the choice of container formats, audio and video compression algorithms, acquisition parameters, and internal file structure. These statistics could then be exploited for several forensic tasks, e.g., for the authentication of digital video files, or the identification of post-processed videos. However,

while providing a pioneering exploration of video container formats from a forensic viewpoint, the work in [33] does not introduce any formal approach to the analysis of file format information, leaving it to visual examination by the analyst.

2.2.3 CODEC features

One of the first steps in video forensics is the analysis of double compression detection, as a mean to integrity verification. The earliest work was carried out by *Wang and Farid* [68] who used the residuals Discrete Fourier Transform (DFT) to uncover trances of *MPEG* double compression. The double compression problem can be analysed taking into account equal or different GOP sizes between the compressions. In the second category, many methods proposed in literature relay on the prediction residual family, as in [68], or the macroblock prediction types family as *Vázquez-Padín's et al.* [65].

A large family of methods has been developed relying on the analysis of DCT coefficients, indeed *Milani et al.* [51] proposed to analyse the distribution of the First Significant Digits (FSD) of DCT coefficients designing a feature based on the Benford's law [11]. *Milani et al.* use the Benford's features to build a set of Support Vector Machine (SVM) classifiers in order to distinguish up to three compression stages in case of H.264 encoded videos. The experiments carried out by the authors in [51] show that the method results in a perfect identification of original compressed videos, whereas a 73% accuracy is obtained for multiple-compression detection, probably due to motion estimation that scrambles the coefficients statistics.

Many recent state-of-the-art contributions [37, 45] focus on peculiarities obtained from macroblock's motion vectors. *He et al.* compute the motion vector field of videos with static backgrounds designing a feature that take into consideration the motion strength of foreground and background regions, which, according to the authors, retains robust footprints of double compression. Furthermore, the authors of [37] study the periodicity of the *modified average prediction residual sequence* upon to post-processing performed by means of the VPF condition [65] and some statistical operators to determine the GOP estimation. The proposed method was evaluated with 6 raw videos in CIF resolution, 352×288 pixels, that are captured with a fixed camera. In these tests, are evaluated double compression performed with $MPEG-4$, and transcoding with AVC, while taking into account Variable Bit Rate (VBR) and Constant Bit Rate (CBR). The results showed in [37]

outperform many state-of-art approaches especially *Vázquez-Padín's* [65], even though the initial assumption on static backgrounds is quite strong, making the technique not applicable in broad spectrum acquisitions scenarios.

Recently, *Bestagini et al.* [15] faced the double compression problem by exploiting the idempotency characteristic of the scalar quantization parameter, extending their work in [13]. The authors argue that the quantization factor is by construction an idempotent operator, and it follows that recompression by the same scalar value will result as the one obtained by applying quantization only once. Thus, the quantization footprint can be used to determine in a recompress-and-observe framework the original codec and GOP. The authors in [15] validated the proposed approach by means of several encoding algorithms such as *MPEG-2*, *MPEG-4* and *AVC*, using 6 raw videos, 11 GOP sizes in the first compression and just one in the second, and 6 quantization parameters (QP) in the second compression. The results show optimum GOP estimations in particular in case of low QP_2 since the second codec does not hide the footprint introduced by the first one. Similar results are obtained for the codec identification, where the best performances are gained when using the *AVC* codec with low QP_2, and the worst in case of the *MPEG-4* codec. One of the main issues that affects *Bestagini et al.*, is the computational cost of such technique, it is clear that in case of up-to-date video resolutions the computational cost of multiple re-encoding is substantial.

Aghamaleki et al. [5, 6] address problems of inter-frame insertion localization, inter-frame deletion localization and double compression detection. They propose an algorithm for forgery detection based on the temporal analysis of residual errors and areas of dominant quantization errors, corresponding to high-textured regions with low motion. The tests are performed on 22 video sequences in CIF and QCIF resolutions encoded in VBR with *MPEG-2*, using the IPPPPPPPPPP GOP structure, obtaining an average detection rate of 92.7%, though if both QPs are small, $QP < 8$, the method proposed by *Aghamaleki et al.* reach an accuracy of 68.18%.

Another interesting contribution to the double compression identification has been published recently by *Chen et al.* [20]. Their work focus on the H.264 codec, uses the distribution of the prediction residual and is applicable to non-aligned GOP structures, meaning that the GOP in the first compression is different than the one in the second compression. *Chen et*

al. argue that the effect on an I-frame encoded as a P-frame increase the prediction residual of the P-frame, thus they determine the *PRED* feature by averaging the prediction residual for non-overlapping 4×4 blocks for each frame, then stating the ability to distinguish between I-frame encoded as a P-frame and a P-frame encoded as a P-frame. In addition they improve the *PRED* feature by evaluating the difference of adjacent frames by means of the Jensen-Shannon divergence and after post-processing a periodic analysis is carried out to determine the first GOP size. The authors in [20] evalute the *PRED* feature using the same dataset of *Vázquez-Padín et al.* [65] but the second compression is performed just with the H.264 codec. The results show comparable results on the compression detection problem, whereas in the GOP estimation they improve [65] with an increment in the range $[2, 10]\%$ depending on video content and bit rate. Moreover *Chen et al.* research was extended in a recent publication by *Zheng et al.* [71], where the innovative element consists of extracting H.264 double compression features from videos with a static background.

It is important mentioning some preliminary works that study the integrity verification problem by means of the recent ITU-T/ISO-IEC coding algorithm, namely HEVC [43]. *Costanzo and Barni* [24] evaluated the double compression detection in case of AVC/HEVC transcoding, being able to determine the value of the quantization parameter used in the first AVC compression by taking into account the idempotency property of the quantization as in [13]. *Huang et al.* [38] evaluates HEVC double compression by means of a co-occurrence matrix based on DCT coefficients, thus building an interesting feature able to reveal alterations carried by double quantization errors. Whereas *Xu et al.* [70] addressed the double compression problem by means of the *SN-PUPM* feature that takes into account the number of prediction units that belong to INTRA, INTER and SKIP modes in I-P frames. Finally, an SVM classifier is trained to label I-P frames and determine the sequence for double compression detection and GOP analysis.

Chapter 3

VISION

In this Chapter we describe the VISION dataset a new contribution to the development of Multimedia Forensics, as benchmark for the exhaustive evaluation of several image and video forensic tools. It is currently composed of 34427 images and 1914 videos, both in the native format and in their social version (Facebook, YouTube and WhatsApp are considered), from 35 portable devices of 11 major brands.

3.1 Introduction

In the last decades, visual data gained a key role in providing information. Images and videos are used to convey persuasive messages to be used under several different environments, from propaganda to child pornography. The wild world of web also allows users to easily share visual contents through social media platforms. Statistics [39] show that a relevant portion of the world's population owns a digital camera and can capture pictures. Furthermore, one-third of the people can go online [36] and upload their pictures on websites and social networks. Given their digital nature, these data also convey several information related to their life cycle (e.g., source device, processing they have been subjected to). Such information may become relevant when visual data are involved in a crime. In this scenario, Multimedia Forensics (MF) has been proposed as a solution for investigating images and videos to determine information about their life cycle [26]. During the years, the research community developed several tools to analyse a digital image,

focusing on issues related to the identification of the source device and the assessment of content authenticity [53].
Generally, the effectiveness of a forensic technique should be verified on image and video datasets that are freely available and shared among the community. Unfortunately these datasets, especially for the case of videos, are outdated and non-representative of real case scenarios. Indeed, most multimedia contents are currently acquired by portable devices, that keep updating year by year. These devices are also capable to acquire both videos and images exploiting the same sensor, thus opening new investigation opportunities in linking different kind of contents [44]. This motivates the need for a new dataset containing a heterogeneous and sufficiently large set of visual data - both images and videos - as benchmark to test and compare forensic tools.
In this Chapter we present a new dataset of native images and videos captured with 35 modern smartphones/tablets belonging to 11 different brands: *Apple*, *Asus*, *Huawei*, *Lenovo*, *LG electronics*, *Microsoft*, *OnePlus*, *Samsung*, *Sony*, *Wiko* and *Xiaomi*. Overall, we collected 11732 native images; 7565 of them were shared through Facebook, in both high and low quality, and through WhatsApp, resulting in a total of 34427 images. Furthermore we acquired 648 native videos, 622 of which were shared through YouTube at the maximum available resolution, and 644 through WhatsApp, resulting in a total of 1914 videos [1].

To exemplify the usefulness of the VISION dataset, we test the performance of a well known forensic tool, i.e., the detection of the Sensor Pattern Noise (SPN) left by the acquisition device [49] for the source identification of native/social media contents; moreover, we describe some new opportunities deriving by the availability of images and videos captured with the same sensor to find a solution to current limits present in the literature. In particular, the proposed dataset contains several devices featuring in-camera digital stabilization, that is known to threaten source identification based on sensor pattern noise. Indeed, in most papers related to SPN [64] [18] [22] [21] digitally stabilized videos are ignored, either by turning the stabilization off or considering non-stabilized devices only. This is unrealistic, considering that most common modern devices (e.g., *Apple* iPhones) are equipped with an in-camera digital stabilization system that cannot be turned off without resorting to third party applications.

[1] Not all the videos were exchanged through social media platforms, more technical details can be found in [59]

The rest of the Chapter is organized as follows: in Section 3.2 a complete description of the VISION dataset is provided for both native and social media contents; in Section 3.3 the dataset is exploited to evaluate some well known forensic applications and to taste new research opportunities. Section 3.4 draws concluding remarks.

3.2 Dataset structure

Images and videos have been acquired from each mobile device by following a specific procedure. First of all, the captured contents refer to the best-quality camera available in the device, in general the one positioned on the upper-rear of the device. Moreover, the devices were configured, when possible, with the highest quality and resolution available (usually the default one for Apple devices, but not necessarily for Android ones).

VISION is mainly thought for video and image source identification applications, as a consequence we organized the data collected from each device into two folders, (see Figure 3.1 for an example), namely:

- *images*: containing native and social exchanged images. We captured images, mainly in landscape-mode, representing flat surfaces (e.g., skies or walls), here defined as *Flat*, and generic images, here defined as *Nat*, for which there are no limitations on orientation or scenario, as it can be seen in Figure 3.2. In addition, the *Nat* images were exchanged via the Facebook and WhatsApp social media platforms.

- *videos*: containing native and social exchanged videos, acquired mainly in landscape-mode. The collected videos represent flat, indoor and outdoor scenarios. The *flat scenario* includes videos belonging to flat surfaces such as walls and skies. The *indoor scenario* comprises videos representing offices or stores, and the *outdoor scenario* contains videos of open areas such as gardens. For each scenario, we used three different acquisition modes: *still-mode*, where the user stands still while capturing the video; *move-mode*, where the user walks while capturing the video; *panrot-mode*, where the user performs a recoding combining a pan and a rotation. Furthermore, the videos belonging to each scenario were exchanged via YouTube and WhatsApp social media platforms.

The structure depicted in Figure 3.1 is maintained also in the naming convention. The contents collected from each device are stored in its root

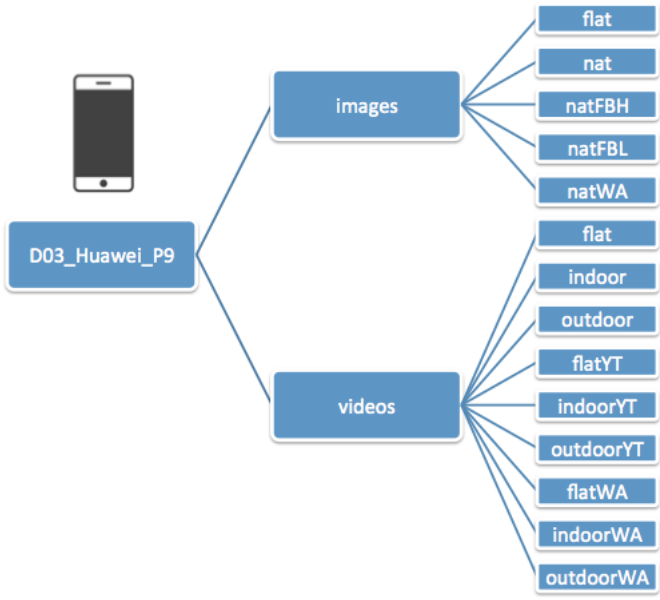


Figure 3.1: VISION folder organization.

folder named `ID_Brand_Model` as in `D01_Samsung_GalaxyS3Mini`. Then, we distinguish between *images* and *videos*, within each of them we have the native content folders and the social ones. A native flat image is called by convention as `ID_I_flat_XXXX.jpg` as in `D01_I_flat_0001.jpg`, where `ID` is the device identifier, `I` identifies it as an image content, `flat` identifies the sub-folder and the type of image, while `XXXX.jpg` is an incremental number. Similarly, the video content naming is `ID_V_scenario_mode_XXXX.mp4` as in `D01_V_flat_panrot_0001.mp4`, where `V` identifies the video content, `scenario` and `mode` refer respectively to the area and the modality of the acquisition procedure. The so described naming convention is also applied to the social folders represented in Figure 3.1: an image uploaded to Facebook in low quality will be named `D01_I_natFBL_0001.jpg`, an image uploaded to Facebook in high quality will be named `D01_I_natFBH_0001.jpg`, while a video exchanged through WhatsApp will be named `D01_V_flatWA_panrot_0001.mp4`.

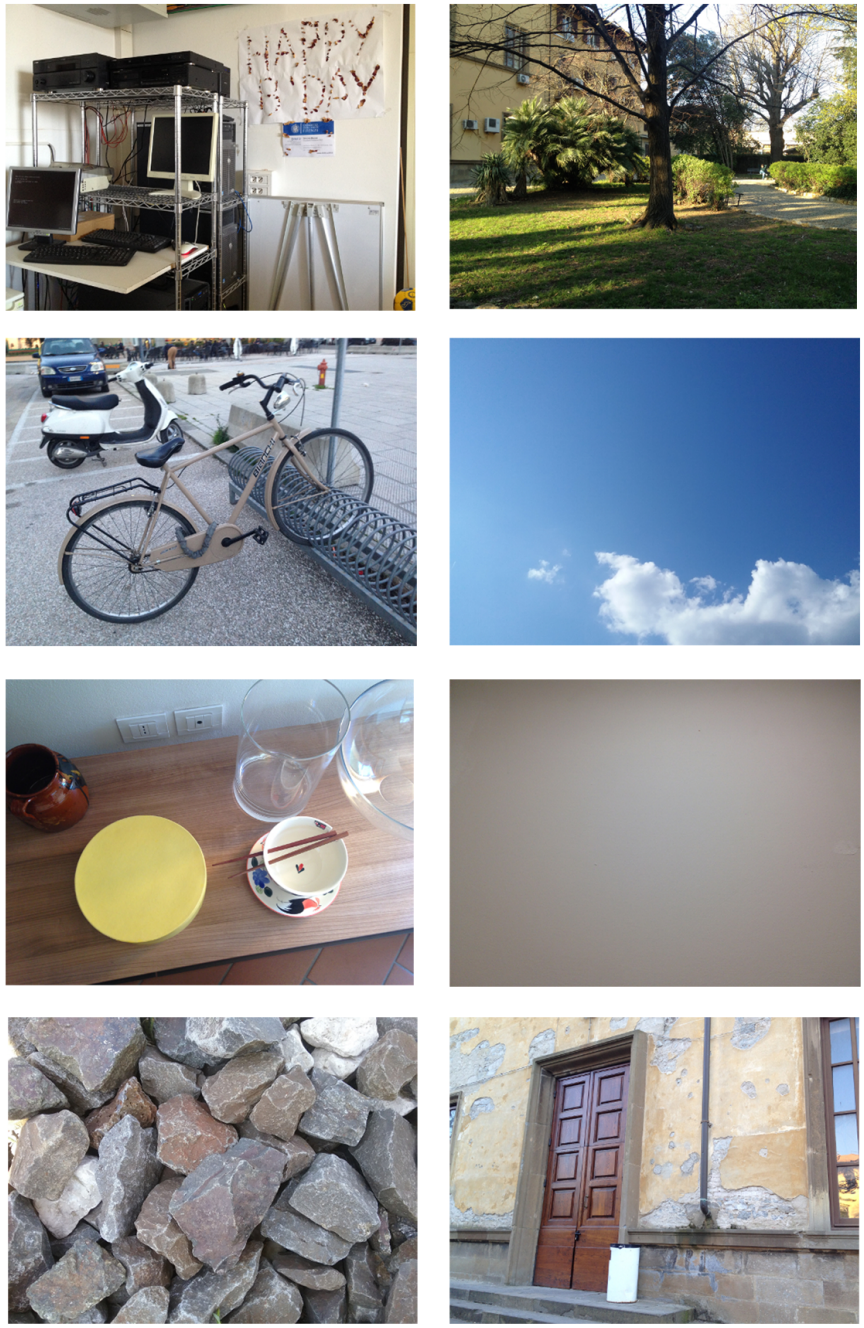


Figure 3.2: Some examples of the images included in the proposed dataset.

Table 3.1: Devices main features. DStab shows the presence or absence of digital stabilization on the acquired content, HDR indicates whether the device supports it, VR stands for video resolution and IR for image resolution.

Brand	Model	ID	DStab	HDR	VR	#Videos	IR	#Images	#Flat	#Nat
Apple	iPad 2	D13	Off	F	1280 × 720	16	960 × 720	330	159	171
Apple	iPad mini	D20	On	F	1920 × 1080	16	2592 × 1936	278	119	159
Apple	iPhone 4	D09	Off	T	1280 × 720	19	2592 × 1936	326	109	217
Apple	iPhone 4S	D02	On	T	1920 × 1080	13	3264 × 2448	307	103	204
Apple	iPhone 4S	D10	On	T	1920 × 1080	15	3264 × 2448	311	133	178
Apple	iPhone 5	D29	On	T	1920 × 1080	19	3264 × 2448	324	100	224
Apple	iPhone 5	D34	On	T	1920 × 1080	32	3264 × 2448	310	106	204
Apple	iPhone 5c	D05	On	T	1920 × 1080	19	3264 × 2448	463	113	350
Apple	iPhone 5c	D14	On	T	1920 × 1080	19	3264 × 2448	339	130	209
Apple	iPhone 5c	D18	On	T	1920 × 1080	13	3264 × 2448	305	101	204
Apple	iPhone 6	D06	On	T	1920 × 1080	17	3264 × 2448	281	149	132
Apple	iPhone 6	D15	On	T	1920 × 1080	18	3264 × 2448	337	110	227
Apple	iPhone 6 Plus	D19	On	T	1920 × 1080	19	3264 × 2448	428	169	259
Asus	Zenfone 2 Laser	D23*	On	F	640 × 480	19	3264 × 1836	327	117	210
Huawei	Ascend G6-U10	D33	Off	T	1280 × 720	19	2448 × 3264	239	84	155
Huawei	Honor 5C NEM-L51	D30	Off	T	1920 × 1080	19	4160 × 3120	351	80	271
Huawei	P8 GRA-L09	D28	Off	T	1920 × 1080	19	4160 × 2336	392	126	266
Huawei	P9 EVA-L09	D03	Off	F	1920 × 1080	19	3968 × 2976	355	118	237
Huawei	P9 Lite VNS-L31	D16	Off	T	1920 × 1080	19	4160 × 3120	350	115	235
Lenovo	Lenovo P70-A	D07	Off	F	1280 × 720	19	4784 × 2704	375	158	217
LG electronics	D290	D04	On	F	800 × 480	19	3264 × 2448	368	141	227
Microsoft	Lumia 640 LTE	D17	Off	T	1920 × 1080	10	3264 × 1840	285	97	188
OnePlus	A3000	D25	On	T	1920 × 1080	19	4640 × 3480	463	176	287
OnePlus	A3003	D32	On	T	1920 × 1080	19	4640 × 3480	386	150	236
Samsung	Galaxy S III Mini GT-I8190	D26	Off	F	1280 × 720	16	2560 × 1920	210	60	150
Samsung	Galaxy S III Mini GT-I8190N	D01	Off	F	1280 × 720	22	2560 × 1920	283	78	205
Samsung	Galaxy S3 GT-I9300	D11	Off	T	1920 × 1080	19	3264 × 2448	309	102	207
Samsung	Galaxy S4 Mini GT-I9195	D31	Off	T	1920 × 1080	19	3264 × 1836	328	112	216
Samsung	Galaxy S5 SM-G900F	D27	Off	T	1920 × 1080	19	5312 × 2988	354	100	254
Samsung	Galaxy Tab 3 GT-P5210	D08	Off	F	1280 × 720	37	2048 × 1536	229	61	168
Samsung	Galaxy Tab A SM-T555	D35	Off	F	1280 × 720	16	2592 × 1944	280	126	154
Samsung	Galaxy Trend Plus GT-S7580	D22	Off	F	1280 × 720	16	2560 × 1920	314	151	163
Sony	Xperia Z1 Compact D5503	D12	On	T	1920 × 1080	19	5248 × 3936	316	100	216
Wiko	Ridge 4G	D21	Off	T	1920 × 1080	11	3264 × 2448	393	140	253
Xiaomi	Redmi Note 3	D24	Off	T	1920 × 1080	19	4608 × 2592	486	174	312

3.2.1 Main features

VISION is composed by 35 mobile devices from low, middle and high price range. There are 13 *Apple* devices, including iPhones and iPads. There are 8 *Samsung* devices including Galaxy phones and tablets. There are 5 *Huawei* and 2 *OnePlus* phones. Furthermore we gathered one device for the following brands: *Asus*, *Lenovo*, *LG electronics*, *Microsoft*, *Sony*, *Wiko* and *Xiaomi*. We collected a few devices of the same brand and model namely: two *iPhone 4S*, two *iPhone 5*, three *iPhone 5c*, two *iPhone 6* and two *Galaxy S3 Mini*. The employed devices had installed the following operating systems: iOS from 7.x to 10.x, Android from 6.x Marshmallow to 7.x Nougat, and the Windows Phone OS 8.1 Update 2.

In Table 3.1 we summarize the main features of the complete Dataset. For each device we report the *Brand*, *Model*, a unique identifier *ID* and the number of collected videos and images with their corresponding resolutions.

In Table 3.1 we also clarify whether videos were captured using in-camera digital stabilization: the reader can see that for most Apple devices if the stabilization is present it is also enabled (the only exceptions are D9 and D13), as it is also for the Sony Xperia, D12. On the contrary, this is not true for all other devices where the in-camera digital stabilization is set off by default. In addition, Table 3.1 clarifies whether the device can acquire images in HDR-*High Dynamic Range* mode: T(*True*) is used if HDR is available and F(*False*) if it is not. Several additional metadata and coding statistics are collected and reported in the Appendix.

We also make available a reduced version of VISION for researchers convenience. This baseline version is composed by 16100 images and 315 videos, both native and social, *equally distributed* among all the devices.

3.2.2 Social contents

The collected contents in VISION were also exchanged through social media platforms; in particular, for images in *Nat* we provide their corresponding uploaded version on Facebook and WhatsApp. We chose to upload only natural images since, from a forensic point of view, having flat surfaces shared through social media is rather unrealistic. In addition, we shared all videos through YouTube and WhatsApp. In the rest of this Section, we explain the procedure used for uploading and downloading media contents through each social media platform.

Facebook Web Platform In order to exchange images via Facebook we created two albums in which we uploaded all images belonging to *Nat* in high and low quality respectively (FBH and FBL from now on), as allowed by the social media platform. Indeed, as deeply explained in [52], these uploading options cause a significantly different compression strategy for the image.

For what concerns the download, we performed single-image downloads and album downloads, although there is no difference between the resulting contents. Album download functionality was recently added to the Facebook website[2] options. The one click album-download button allows downloading a zip version of each album; in each zip-file, the images are renamed by an incremental number as: 1.jpg, 2.jpg, ... n.jpg , where n is the number of images in the folder.
Since the collection of VISION lasted over a year, we exchanged data both before and after this update. We took care to provide a matching naming between the original content and the social media one: we used the SSIM index [69] as a metric to determine whether the two images depict the same content. Consequently, if the native image name is `D01_I_nat_0001.jpg`, its Facebook high quality counterpart will be named `D01_I_natFBH_0001.jpg`.

YouTube Web Platform All video contents were uploaded to YouTube with the *Public privacy* flag and collected into a playlist. During the collection of VISION we exploited different solutions to speed-up the downloading process, but maintaining the constraints of highest resolutions and no download compression. We encountered two software solutions to accomplish this goal, namely *ClipGrab*[3] and *Youtube-dl*[4]. Both software are freely available and can be used on several operating systems such as Unix and Windows. The main difference between the two is that the *ClipGrab* GUI can download one video at a time, while the *youtube-dl* command line can download also playlists. As an example, we provide the following *youtube-dl* command line call to download a playlist:[5].

```
youtube-dl -f 137+140/bestvideo+bestaudio -o "%(title)s.%(ext)s"
--yes-playlist "device_url_playlist"
```

The options after the **-f** refers to the quality of video resolution and audio

[2]Facebook website on March 2017.
[3]ClipGrab v3.6.3 - `www.clipgrab.org`
[4]youtube-dl v2017.03.10 - `http://rg3.github.io/youtube-dl/`
[5]We recommend downloading less than 20 videos at a time due to the YouTube policy.

settings; here the meaning is to choose the highest video resolution and audio quality, if not available choose the second-best pair and so on from left to right. Then with option **-o** we set the output video name and extension to be the YouTube video name and the default extension, i.e. mp4. For the complete documentation we advise the reader to refer to [16].

Similarly to the image naming convention, a video recorded in an outdoor scenario with a panrot movement has the following name: `D01_V_outdoor_panrot_0001.mp4`, while its YouTube counterpart will be named `D01_V_outdoorYT_panrot_0001.mp4`.

WhatsApp Mobile Application: All native video contents and images belonging to *Nat* were exchanged via WhatsApp v2.17.41 using an *iPhone 7* A1778 with iOS v10.3.1. We decided to use the mobile-application instead of the desktop one since the latter does not compute any compression to the shared file, while the mobile one does so. We used an iPhone since it produces a media file that is less compressed than the Android one, due to WhatsApp implementation choices. In this way, we provided an equilibrate spectrum of social image contents qualities: namely high and low provided by Facebook, and medium from WhatsApp. As to the naming convention, for these files we had the same issue as in Facebook: since downloaded images are renamed, we matched images using the SSIM index.

The videos downloaded from WhatsApp follow the same name structure, (e.g., `D01_V_outdoorWA_panrot_0001.mp4`.

3.3 Forensics applications

This dataset was created to provide a benchmark for the forensic analysis of images and videos. In this Section, we exploit all the collected contents to test the source identification technique based on the Sensor Pattern Noise. In this scenario, the aim is to identify the source of an image or video by evaluating the correlation between the SPN fingerprint estimated from the investigated content, and the device reference fingerprint, computed from a set of images or a video taken by this device.

We tested different application scenarios:

- Image Source Identification (ISI), where a query image is matched with a device reference computed from a set of images taken by the device;
- Video Source Identification (VSI), where a query video is matched with

a device reference computed from the frames of a video taken by the device.

The identification is performed according to the classical workflow [19]: a camera fingerprint $\mathbf{K}$ is estimated from N still images or video frames $\mathbf{I}^{(1)}, \ldots, \mathbf{I}^{(N)}$ captured by the source device. A denoising filter [49], is applied to each image/frame and the noise residuals $\mathbf{W}^{(1)}, \ldots, \mathbf{W}^{(N)}$ are obtained as the difference between each frame and its denoised version. Then, the camera fingerprint estimate $\widetilde{\mathbf{K}}$ is derived by the maximum likelihood estimator [19]:

$$\widetilde{\mathbf{K}} = \frac{\sum_{i=1}^{N} \mathbf{W}^{(i)} \mathbf{I}^{(i)}}{\sum_{i=1}^{N} (\mathbf{I}^{(i)})^2}. \tag{3.1}$$

The fingerprint of the query is estimated in the same way by the available image or video frames. Then, the Peak to Correlation Energy (PCE) between the reference and the query pattern is computed and compared to a threshold [35]: if the PCE is higher than the threshold, then it is decided that the query content has been acquired by the reference device.

3.3.1 Image Source Identification

In this scenario, the reference SPN for each device is estimated using 100 still flat field images. Then, we run four experiments using natural, WhatsApp, Facebook high-quality, and Facebook low-quality images as queries. In all experiments, we consider for each device 100 matching cases (images from the same device) and the same number of mismatching cases (images randomly chosen from other devices). The achieved results are reported using ROC curves, that plot true positive rate against false positive rate (see Figure 3.3). The overall performance are summarized in Table 3.2 where, for each experiment, we also reported the dataset path of the query images and the Area Under Curve. `ID_Brand_Model` stands for any of the available device e.g., `D03_Huawei_P9.`

3.3.2 Video Source Identification

Here, the source of a test video is determined based on references estimated from a flat-field video. In particular, the reference SPN for each device is estimated from the first 100 frames of a flat video. Then, three experiments are performed using natural, YouTube and WhatsApp videos as queries,

Table 3.2: Performance of Image Source Identification in growing difficulty scenarios.

Experiment	**Test Path**	**AUC**
1	ID_Brand_Model/images/nat	0.9906
2	ID_Brand_Model/images/natWA	0.9860
3	ID_Brand_Model/images/natFBH	0.9859
4	ID_Brand_Model/images/natFBL	0.9544

Table 3.3: Dataset paths for VSI experiments.

Experiment	**Test path**	**AUC**	
		All Videos	Unstab. Videos
1	ID_Brand_Model/videos/flat ID_Brand_Model/videos/indoor ID_Brand_Model/videos/outdoor	0.7069	0.9394
2	ID_Brand_Model/videos/flatYT ID_Brand_Model/videos/indoorYT ID_Brand_Model/videos/outdoorYT	0.6032	0.7700
3	ID_Brand_Model/videos/flatWA ID_Brand_Model/videos/flatWA ID_Brand_Model/videos/flatWA	0.5262	0.5437

respectively. The fingerprint of each tested video is estimated from the first 100 frames. We consider for each device all available matching cases (videos from the same device) and the same number of mismatching cases (videos randomly chosen from other devices). The achieved results are reported in Figure 3.4, where only non-stabilized cameras are analysed, and in Figure 3.5, where all devices in the dataset are considered. This experiment shows that performance of VSI strongly drop when digitally stabilized videos are involved. In Table 3.3 we briefly summarize for each test the paths in the dataset of tested videos and the Area Under Curve values obtained with and without stabilized videos.

For in-camera stabilized videos, possible solutions are still under development, as the one proposed in [63]. Anyway the solution in [63] is proved to be effective only on third party (out-camera) digital stabilization (*FFmpeg*), and when a non-stabilized video is available as reference. Unfortunately, most of the considered devices enforce in-camera digital stabilization, without an option to turn it off in the standard camera application.

3.3.3 Image vs Video SPN fingerprint

In the research community, ISI and VSI applications are separately studied, so that there's still no better way to perform image and video source identification for the same device than computing two different reference SPNs, one for still images and one for videos, respectively.

A first step towards an integration of these cases is a hybrid source identification (HSI) approach, that exploits still images for estimating the fingerprint that will be used to verify the source of a video, as proposed in [44]. Authors of [44] investigate the geometrical relation between image and video acquisition processes. Indeed, even if the sensor is the same, videos are usually acquired at a much lower resolution than images: top-level smartphones reach 4K video resolution at most (8 Megapixels per frame), but can easily capture 20 Megapixels images. To achieve that, in video recording a central crop is carried so to adapt the sensor size to the desired aspect ratio (commonly 16:9), then the selected pixels are scaled to match the desired video resolution. As a direct consequence, the fingerprints extracted from images and videos cannot be directly compared and most of the times, because of cropping, it is not sufficient to just scale them to the same resolution. Instead, image-based and video-based fingerprints are linked by the cropping and scaling factors between image and video sensor portion, that usually change across different device models.

With the aim of facilitating researchers exploring the HSI framework within the VISION dataset, we provide the cropping and scaling factor for several devices contained therein. For simplicity, we limit to non-stabilized devices; the hybrid analysis for stabilized devices is even more complex, and is one of the future research scopes this dataset has been built for. In order to estimate cropping and scaling factors, for each device we estimated the video and image references from the videos contained in `ID_Brand_Model/videos/flat` and from the images in `ID_Brand_Model/images/flat`, respectively. Specifically, we estimated each image reference fingerprint from 100 flat field images and each video reference fingerprint from 100 frames of a flat field video. The cropping and scaling factors are estimated by a brute force search, as suggested in [34]. In Table 3.4 we report the scaling factor and the corresponding cropping corner (upper-left corner along x and y axes) yielding the maximum PCE for each examined device. We consider the parameter search unsuccessful if the obtained maximum PCE is lower than 50 (denoted by "n.a." in Table 3.4). For instance, with the device $D11$ an image finger-

Table 3.4: Estimated Cropping and Scaling factors for non stabilized videos

ID	D01	D03	D07	D08	D09	D11	D13	D16	D17	D21
Scaling	0.5	0.48	0.27	1	0.61	0.59	1	0.46	0.59	n.a.
Cropping [**x y**]	[0 228]	[0 372]	[0 7]	[408 354]	[227 411]	[0 307]	[-160 0]	[8 396]	[0 1]	n.a.

ID	D24	D26	D27	D28	D30	D31	D32	D33	D35	D22
Scaling	0.5	n.a.	0.5	0.36	0.47	0.46	0.59	0.52	0.39	0.49
Cropping [**x y**]	[0 240]	n.a.	[0 228]	[0 0]	[39 10]	[9 397]	[0 0]	[464 693]	[0 306]	[0 246]

print should be scaled by a factor 0.59 and then cropped on the upper left side of 307 pixels along the y axis to match the video fingerprint (the right and down cropping are derived by the corresponding video size). $D13$ is a pretty unique case in which the full frame is applied for videos and the left (and right) cropping of 160 pixels is applied to capture images. We put a -160 meaning that the video frame is cropped by 160 pixel to capture images. Finally, we notice that we were not able to register the fingerprints for the devices $D21$ and $D26$ by means of the presented techniques. A deeper analysis of the registration techniques is still an open topic.

3.4 Remarks

In this Chapter we propose a new image and video dataset useful for benchmarking multimedia forensic tools. We collected thousands of images and videos from portable devices of most famous brands, including those featuring in-camera digital stabilization. We also prepared the "social version" of most contents, by uploading and downloading them to/from well known social media platforms, namely Facebook, YouTube and WhatsApp.

In addition showed examples of some popular applications that would benefit from the proposed dataset, such as the Video Source Identification, for which there are no sizeable benchmarks available in the research community. Furthermore we showed how this dataset allows the exploration of new forensic opportunities such as comparing camera reference fingerprints estimated from still images and from videos. The whole dataset is made available[6] to the research community, along with a guide that clarifies its structure and several csv files containing technical information.

[6] Vision is available online at `https://lesc.dinfo.unifi.it/en/datasets`.

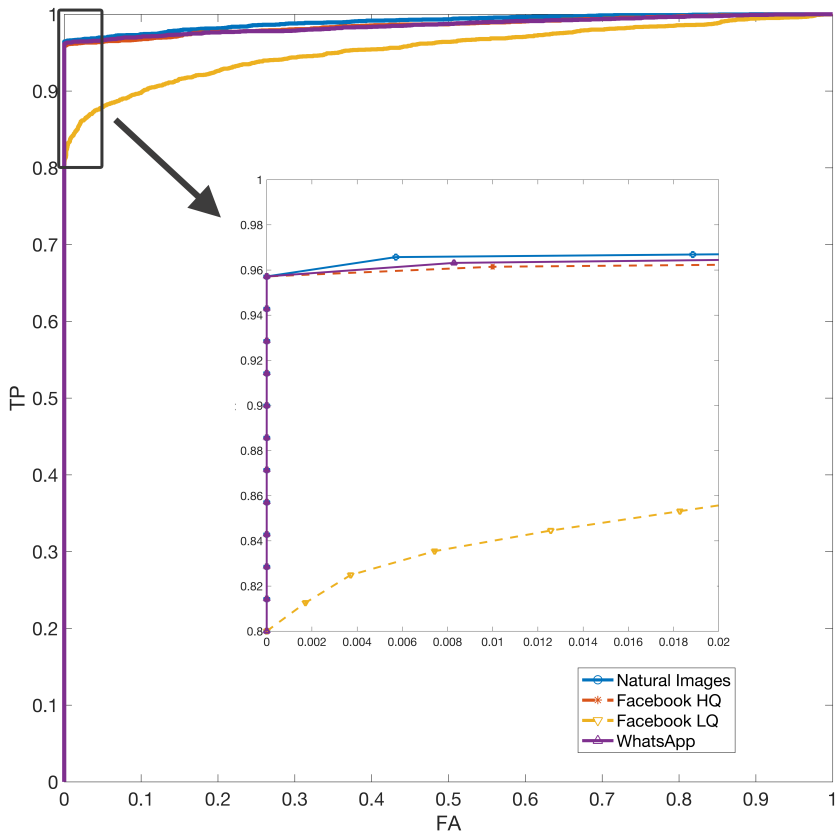


Figure 3.3: ISI performance on Native, Facebook (HQ and LQ) and WhatsApp images using flat field references.

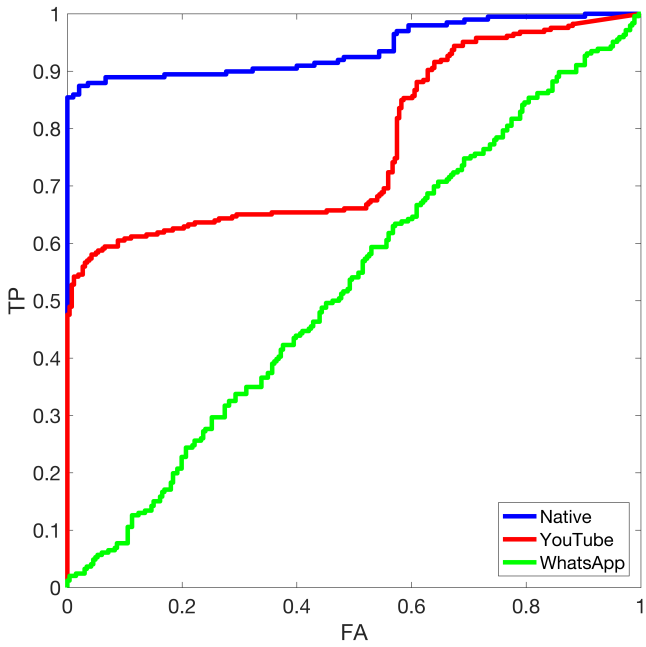


Figure 3.4: The VSI performance on Native, YouTube and WhatsApp videos (in blue, red and green respectively) considering only devices devices without in-camera digital stabilization.

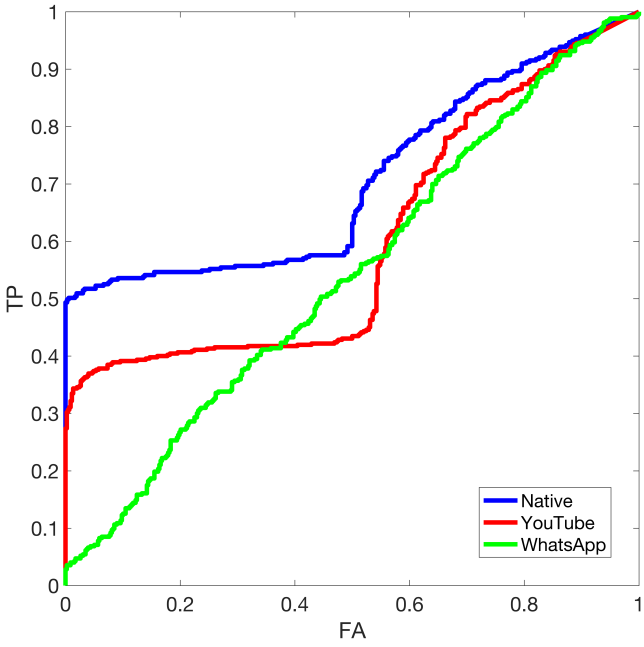


Figure 3.5: The VSI performance on Native, YouTube and WhatsApp videos (in blue, red and green respectively) considering all available devices.

Although VISION is a huge collection of media contents, we believe that there is space for future improvements, indeed we are currently working to extend VISION with more videos, by means of a Mobile Application (MOSES [58]). MOSES allows everyone with a smart device, both Android and iOS based, to contribute to the proposed Dataset. The end-user can record a video at default settings, under different scenarios and motions, as in VISION guidelines, then upload it to our servers. Each acquisition has a maximum duration of 30 seconds, since we have to consider the user bandwidth and storage limits, despite that, we considered it sufficient since most mobile devices record videos at a frame rate of 30 frame per second, resulting in the availability of video sequences composed by 900 frames. In addition, MOSES collects further information available from the camera, that are related to the source and the video file format, such as GPS coordinates or the container type. It is important to notice that the acquisition process is carried out by MOSES using the native-camera application of each device, while uploading videos from the device gallery is not allowed. This choice aims at preventing the upload of manipulated videos, or videos that have not been captured by the device running the application. MOSES is still under development since many issues have to be faced, how to properly store the captured contents but also accessing more and more information from the original device.

Chapter 4

Low-Level features analysis

In this Chapter we focus on the analysis of video file containers and, based on its fragility and variability between sources, we introduce two main forensic applications. The first focuses on the integrity verification based on the dissimilarity between two video containers structure; the latter focuses on the identification and classification of the video brand/model based on the analysis the containers structure and content. This last application rests on the likelihood-ratio framework, more and more approved by the forensic community as an appropriate way to exhibit findings in court.

4.1 Introduction

Multimedia Forensics has designed several tools to blindly deduce the digital history of a digital video, based on the fact that any step in the media life cycle leaves distinctive traces on the content itself.

In this regard, the techniques for the forensic analysis of digital videos mainly focused on the data stream, i.e. the audio-visual signal, based on the detection of artefacts and inconsistencies in the statistics of the content itself. A less explored approach is the analysis of the file formats and metadata, to determine their compatibility, completeness, and consistency with respect to the context in which it is assumed the resource has been created. Indeed, the potentiality of the information contained within the video file containers has not still been deeply investigated; naturally the video container structure

and content can be extremely variegated and customized among different sources. This fact complicates the analysis, but at the same time it makes possible a higher discriminating power of the discovered peculiarities.

The goal of this Chapter is to bring a contribution to the field of video integrity verification. First, expanding Gloe et al. idea on the importance of video file container analysis, we provide a formal measure allowing to quantify the distance between two containers. The measure accounts for both the container structure and content, thus providing an excellent capability in distinguishing videos whose integrity is preserved from videos whose integrity is compromised. Then, building on this measure we define a formal framework for container- based brand/model identification and classification: noticeably, this framework analyses the container structure and content of training data and automatically computes weighting factors telling which parts of a container are mostly relevant for discriminating a specific brand/-model from other. Moreover, the framework expresses results in terms of likelihood-ratio, which is considered the best way for evaluative reporting in the forensic field [50], [7].

In detail, we introduce two main video forensic applications based on the analysis of the file container:

- the video integrity verification based on the measure of the dissimilarity between file containers structures;
- a brand/model identification and classification, based on the containers structure and content, properly modelled in the likelihood ratio framework. The identification task consists of checking the compatibility degree of a query video with an alleged source brand/model; The classification task automatically assigns the most likely source brand/-model from a set of known brands/models.

Concerning the first application, we have proved the effectiveness of the proposed approach under three different challenging scenarios: a video has been exchanged through YouTube and WhatsApp; a video has been cut, without further encoding, through FFmpeg[1]; a few video metadata have been changed through ExifTool[2].Concerning the second application,

[1]FFmpeg is a free software project that produces libraries and programs for handling multimedia data (available at: `https://www.ffmpeg.org/`).

[2]ExifTool is a platform-independent Perl library plus a command-line application for reading, writing and editing meta information in a wide variety of files (available at: `http://www.sno.phy.queensu.ca/~phil/exiftool`).

the method is proved to be effective for the identification/classification of both brand and model of portable devices. More specifically, both applications have been tested in a realistic scenarios, by exploiting the newly released VISION Dataset [59] presented in detail in Chapter 3.

The rest of the Chapter is organized as follows: Section 4.2 introduces the standards related to the most common video file formats, mainly *MP4* and *MOV*, and briefly summarises the structure of a video container; Section 4.3 formalizes the mathematical framework and the metrics exploited to compare the structure and the contents of video file containers; Section 4.4 is dedicated to the experimental validation of the proposed technique; finally, Section 4.5 draws some final remarks and outlines future works.

4.2 ISO container structure

The main features of *MP4 File Format* [3] and *MOV File Format* [10] containers derive from the standard ISO/IEC 14496 Part 12 [4]. The *ISO Base format* is characterized by a sequence of atoms or boxes. Each atom consists of a header and a data box, and occasionally nested atoms; as an example, in Figure 4.1 a common structure of an MP4-like container is reported. The file type box, `ftyp`, is a four letter code that is used to identify the type of encoding, the compatibility or the intended usage of a media file. According to the latest ISO standards, it is considered as a semi-mandatory atom, i.e. the ISO expects it to be present and to be explicit as soon as possible in the file container. In the example given in Table 4.1, the fields of the `ftyp` descriptor explain that the video file is MP4-like and it is compliant to the MP4 Base Media v1 [IS0 14496-12:2003] (here *isom*) and the 3GPP Media (.3GP) Release 4 (here *3gp4*) specifications[3]. While in the movie box, `moov`, there is a nested atom which contains in its sub-structure the metadata needed for the decoding of the data stream contained in the following `mdat` atom. It is important to note that `moov` may contain multiple instances of the `trak` box, as reported in Figure 4.1. The `trak` atom is mandatory and its cardinality depends on the number of streams contained in the media; for example, if the video contains a visual-stream and an audio-stream, there will be two independent `trak` atoms.

All the mentioned container structure and data are extracted from the

[3]We suggest the reader to look at `http://www.ftyps.com/` to have a better understanding of the values reported in Table 4.1.

Table 4.1: Example of the `ftyp` atom descriptor belonging to a video acquired with a *Samsung Galaxy S3*.

Name	Value
majorBrand	isom
minorVersion	0
compatibleBrand_1	isom
compatibleBrand_2	3gp4

video by means of the MP4 Parser library [9], and stored in an XML file for further analysis, as explained in the following Section.

4.3 Forensic Analysis of Video File Container

The video container will be represented as a labelled tree where internal nodes are labelled by atoms names (e.g. *moov-2*) and leaves are labelled by field-value attributes (e.g. *@stuff: MovieBox[]*). To take into account the order of the atoms, each XML-node is identified by a 4-byte code of the corresponding atom along with an index that represents the relative position with regards to the other siblings at a certain level. On the contrary, in our representation the order of field-values, within the same atom, is not considered, since the analysis demonstrated that this sub-ordering is not discriminative.

Given a video X, its container is represented as an ordered collection of atoms $a_1, \ldots, a_n$, possibly nested. Each atom can then be described in terms of a set of field-value attributes, as $a_i = \big(\omega_1(a_i), \ldots, \omega_{m_i}(a_i)\big)$. By combining the two previous descriptions, the video container can be characterised by a list of field-value attributes $X = \big(\omega_1, \ldots, \omega_m\big)$.

To each field-value attribute $\omega \in X$, we also associate the path $p_X(\omega)$, that is the ordered list of atoms to be crossed to reach ω in X starting from the root. As an example, consider the videos X and $X^{'}$ reported in Figure 4.1 and 4.2 respectively. The path to reach the field-value $\omega =$ *@timescale: 1000* in X is the sequence of atoms (*ftyp-1, moov-2, mvhd-1*). The same field-value is available in $X^{'}$ but $p_{X'}(\omega)$ is given by the list (*ftyp-1, free-2, mdat-3, moov-4, mvhd-1*). In this sense, we can state that $p_X(\omega) = p_{X'}(\omega^{'})$ if the same ordered list of atoms is crossed to reach the field-values in the two trees respectively.

In summary, the video container structure will be described by a list of field-value attributes $X = (\omega_1, \ldots, \omega_m)$, and their corresponding paths $p_X(\omega_1), \ldots, p_X(\omega_m)$.

In the following we will formally define two metrics to be used for the forensic analysis of video file containers. More specifically, the first one is thought for video integrity verification, while the second one is applicable for the brand/model identification and classification. From now on, we will denote with $\mathcal{X} = \{X_1, \ldots, X_N\}$ the world set of digital videos, and $\mathcal{C} = \{C_1, \ldots, C_s\}$ the set of disjoint possible origins, e.g., device Huawei P9, iPhone 6s, etc. .

4.3.1 Video Integrity Verification

When a video is processed in any way (with respect to its native form), its integrity is compromised [62]. In particular, any processing, even without further encoding, strongly alters the container structure with respect to its native structure. Conversely, the file containers of native contents generated from a specific source device are expected to have a small intra-variability. As an example, in Figure 4.1[4] and 4.2, we provide two file containers in comparison, one for a native video coming from a Samsung Galaxy S3 and one from the same video after editing with FFmpeg. It is indisputable that the differences in the position of the `moov` and `mdat` boxes can be exploited to determine that the two file containers come from videos with different processing histories, even without taking into account the atoms contents. Thanks to these traces, given a video X whose integrity has to be assessed, and a native video $X^{'}$ coming from the same device, their container structure dissimilarities can be exploited to expose evidences of integrity violation, as follows.

Given two containers $X = (\omega_1, \ldots, \omega_m), X^{'} = (\omega_1^{'}, \ldots, \omega_m^{'})$ with the same cardinality m[5], we define a similarity core function between two field-values as

$$S(\omega_i, \omega_j^{'}) = \begin{cases} 1 & \text{if } \omega_i = \omega_j^{'} \text{ and } p_X(\omega_i) = p_{X'}(\omega_j^{'}) \\ 0 & \text{otherwise} \end{cases} \tag{4.1}$$

We can easily extend it to the comparison between a single field-value $\omega_i \in X$

[4]The complete XML structure is given in Appendix A.2.

[5]if the two containers have different cardinality, we pad the smaller one with empty field-values to obtain the same value m.

Figure 4.1: A fragment of the Video container original content of a *Samsung Galaxy S3* device.

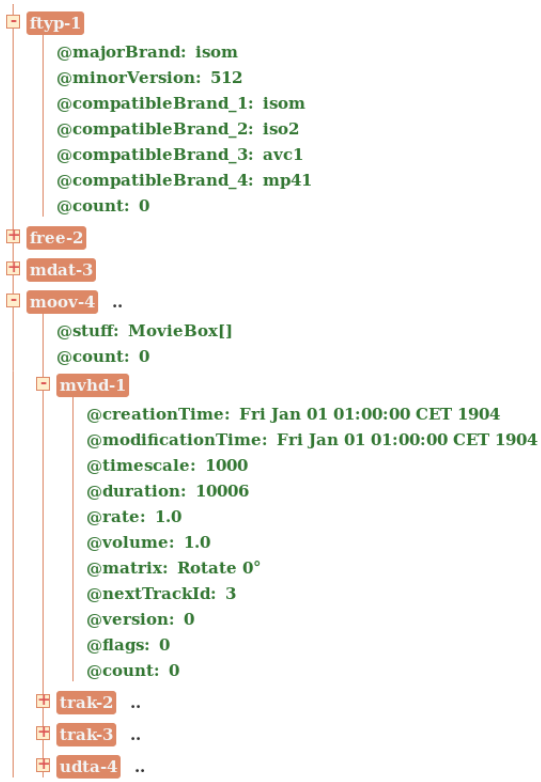


Figure 4.2: A fragment of the Video container FFmpeg cut from a *Samsung Galaxy S3* device.

and the whole $X^{'}$ as

$$\mathbf{1}_{X'}(\omega_i) = \begin{cases} 1 & \text{if } \exists\, \omega_j^{'} \in X^{'} : S(\omega_i, \omega_j^{'}) = 1 \\ 0 & \text{otherwise} \end{cases} \tag{4.2}$$

Then, the dissimilarity between X and $X^{'}$ can be computed as the mismatching percentage of all field-values, i.e.,

$$mm(X, X^{'}) = 1 - \frac{\sum_{i=1}^{m} \mathbf{1}_{X'}(\omega_i)}{m} \tag{4.3}$$

and, to preserve symmetry, the degree of dissimilarity between X and $X^{'}$ can be computed as

$$D(X, X^{'}) = \frac{mm(X, X^{'}) + mm(X^{'}, X)}{2}. \tag{4.4}$$

4.3.2 Video Brand/Model identification and classification

In this Section we formally define an approach that, given a query video, outputs the likelihood that the video belongs to an alleged brand/model. This approach can be used directly for video brand/model identification (that is, checking the compatibility degree of a query video with an alleged source brand), and it can be used as a building block for brand/model classification (that is, automatically assign the most likely source brand from a set of known brands) Overall, we denote Ω the set of all fields-values available in the world set, i.e.,

$$\Omega = \left\{ \omega_1, \ldots, \omega_M | \ \forall i \ \omega_i \in X_j, \exists j \right\}$$

Given a class $C \in \mathcal{C}$, we can split the world set in two subsets $\mathcal{X}_C \cup \mathcal{X}_{\overline{C}}$ where $\mathcal{X}_C = \{X_1, \ldots, X_{N_C} \in C\}$ and $\mathcal{X}_{\overline{C}} = \mathcal{X} \setminus X_C$. Then, for each field-value $\omega \in \Omega$, we determine its discrimination powers for the two classes, namely

$W_C(\omega)$ and $W_{\overline{C}}(\omega)$, based on its frequency in $\mathcal{X}_C$ and $\mathcal{X}_{\overline{C}}$ respectively:

$$W_C(\omega) = \frac{\sum_{X_i \in \mathcal{X}_C} \mathbf{1}_{X_i}(\omega)}{N_C} \tag{4.5}$$

$$W_{\overline{C}}(\omega) = \frac{\sum_{X_i \in \mathcal{X}_{\overline{C}}} \mathbf{1}_{X_i}(\omega)}{N_{\overline{C}}} \tag{4.6}$$

where $N_{\overline{C}}$ is the cardinality of $\mathcal{X}_{\overline{C}}$.
We now formalize how to assess whether a query video $X = \{\omega_1, \dots \omega_m\}$ belongs to a class C. The problem is defined as a two hypotheses test

$$H_0 : X \in \overline{C}$$
$$H_1 : X \in C$$

For each field-value $\omega_i \in X$ we compute its likelihood ratio by approximating the conditioned probability with the conditioned frequency computed by Eq 4.5:

$$P(\omega_j|H_0) = W_{\overline{C}}(\omega_j)$$
$$P(\omega_j|H_1) = W_C(\omega_j)$$

Then, supposing that ω_j are independent, the log likelihood ratio of X for the class C is computed as

$$L_C(X) = \log \prod_{\omega_j \in X} L_C(\omega_j) = \log \prod_{\omega_j \in X} \frac{W_C(\omega_j)}{W_{\overline{C}}(\omega_j)}. \tag{4.7}$$

This approach allows to produce for a query video X a sequence of likelihood ratios, one for each of the considered classes, i.e., $X \to (L_{C_1}(X), \dots, L_{C_s}(X))$.

Note that the assumption of independent ω_j may be restrictive and unsatisfied in some cases. For instance some field-values may be repeated several times within the same path of the file container. In this case, equation 4.7 may produce biased ratios, pushing the likelihood towards the class C or the class $\overline{C}$ improperly. At the same time, it is not feasible to study all possible dependencies between different field-values. We discuss this issue and propose a way to reduce its influence in the Appendix A.1; however, the independence assumption proved to have a limited impact during experimental validation.

4.4 Experimental validation

In this Section we show how the proposed methods can be effectively exploited to perform video integrity verification, brand/model video identification and classification.

We tested the proposed techniques on a subset of the VISION dataset [59], composed of 31 portable devices of 8 major brands[6], that leads to a collection of 578 videos in the native format and in their corresponding social version (YouTube and WhatsApp are considered).

The available data, represented in Table 4.10, includes both videos captured using *Android*, which uses the MP4 [3] file format, and videos captured using *iOS*, which are stored in MOV file format [10]. A detailed description of the dataset is given in [59].

4.4.1 Video Integrity Verification

We considered four different scenarios of integrity violation:

- *WhatsApp*: the video is exchanged through the WhatsApp[7] social platform, that performs a strong modification of both data stream and file container structure (the video is re-encoded and, possibly, downscaled);
- *YouTube*: the video is exchanged through the YouTube[8] social platform; also in this case, a strong modification of both data stream and file container structure occurs;
- *FFmpeg*: the video is cut after 10 seconds using FFmpeg, but without re-encoding[9];
- *Exiftool*: only date-related metadata are changed using Exiftool[10].

[6] VISION dataset is composed by 35 devices; however, four of them, namely D07, D12, D17 and D35, have been excluded since the *MP4 parser* software was not able to correctly extract the container structure.

[7] The videos were uploaded/downloaded via an iPhone7 A1778 with iOS v10.3.1 from the corresponding WhatsApp mobile application v2.17.41.

[8] The videos were uploaded via the YouTube web interface and downloaded at the maximum resolution available with ClipGrab [1].

[9] To cut a video without re-encoding we adopted the following FFmpeg command line call: `ffmpeg -i input.mp4 -ss 00:00:00 -t 00:00:10 -acodec copy -vcodec copy output.mp4`

[10] The adopted Exiftool command line call to modify all the date related fields to a specific time is `exiftool "-AllDates=1986:11:05 12:00:00" "input.mp4"`

For each of the four cases, tests are performed as in the following. We consider the set of videos available for each class device C, namely $X_1, \ldots, X_{N_C}$, and we compute the intra-class dissimilarities $D_{i,j} = D(X_i, X_j), \forall i \neq j$ based on equation (4.4), where both reference and query videos are in their original native version. For simplicity we denote with D_{oo} the intra-class set of dissimilarities. Then, we consider the corresponding $D^t_{i,j} = D(X_i, X^t_j)$, $\forall i \neq j$, where X^t_j is the altered version of the video X_j with the tool-t (i.e., WhatsApp, YouTube, FFmpeg, or Exiftool). We denote with D^t_{oa} the inter-class set of dissimilarities. By applying this procedure to all the considered devices, we collected 2890 samples for both D_{oo} and any of the four D^t_{oa}. We report in Table 4.2 the maximum D_{oo} and the minimum of D^t_{oa} for each of the tested cases, indeed, the following relations hold:

$$\begin{aligned}
&\max D_{oo} < \min(D^{WhatsApp}_{oa}),\\
&\max D_{oo} < \min(D^{YouTube}_{oa}),\\
&\max D_{oo} < \min(D^{FFmpeg}_{oa}),\\
&\max D_{oo} > \min(D^{Exiftool}_{oa}).
\end{aligned}$$

These values demonstrate that the exchange through *WhatsApp* and *YouTube* strongly compromises the container structure, and thus it is easily possible to detect the corresponding integrity violation. Interestingly, cutting the video using *FFmpeg*, without any re-encoding, also results in a strong container modification that can be detected. On the contrary, the date modification with *Exiftool*, induced on containers a modification that is comparable with the observed intra-variability of native videos for some brand, model and device configuration, and thus it is not detected.

In order to investigate more accurately the performance, we separately analysed the results obtained for each device. Thus, for each device class, we report in Table 4.3 the minimum and maximum intra-class variability on the original videos; next, for each of the four scenarios, the minimum and maximum variability between original and altered videos, and the area under the receiver operating characteristic curve (AUC), summarizing the discrimination ability between original and altered contents (where a value equal to one denotes a perfect classifier).

As expected, we have a unitary AUC for all *WhatsApp*, *YouTube* and *FFmpeg* tests. As to *Exiftool*, we notice that fourteen devices still yield unitary AUC, eight more devices yield an AUC greater than 0.8, and the remaining nine devices yield AUC below 0.8, with the lowest value of 0.593

Table 4.2: Intra-Class and Inter-Class dissimilarities for each considered scenario

$max\,(D_{oo})$	$min\,(D_{oa}^{WhatsApp})$	$min\,(D_{oa}^{YouTube})$	$min\,(D_{oa}^{FFmpeg})$	$min\,(D_{oa}^{Exiftool})$
0.328	0.814	0.790	0.662	0.000

obtained for D20 (*Apple* iPad mini running iOS 8.4). Noticeably, all devices yielding an AUC lower than 1 belong to the *Apple* or *Xiaomi* brands. Indeed, as shown in the *Original* column of Table 4.3, the intra-class variability for these devices is noticeably higher than for the others.

Summarizing, for the video integrity verification tests we obtain perfect discrimination for videos altered by social network or FFmpeg, while for Exiftool we obtain an AUC greater than 0.82 on 70% of the considered devices.

It can be noted that, using state of the art methods based on data stream analysis, it would be nearly impossible to detect cutting with *FFmpeg*, since no re-encoding occurs; still, cutting an arbitrary portion of the video can be considered a realistic and powerful attack.

Finally, let us highlight that the comparison between a video reference and a video query required on average just 0.15 seconds.[11] For this reason, we believe that this contribution to video integrity verification is highly valuable.

4.4.2 Video Brand/Model Identification and Classification

In this Section we evaluate the performance of the proposed method using a subset of the VISION dataset, composed of videos belonging to devices of 8 different brands for a total of 25 models, to identify and classify the brand/model of a query video content.

Brand/Model Identification

To test brand identification, we consider all available videos from brand C: for each $X_i \in C$ we randomly choose an $Y_i \in \overline{C}$. Then, weights (equation 4.5) are computed using all videos from class C, except for X_i, and an equal number of videos from other brand classes. The likelihoods $L_C(X_i)$ and

[11] The computational cost has been computed on an Intel(R) Core(TM) $i7-3770$ CPU at 3.40GHz, running all algorithms by means of Python 2.7.

Table 4.3: Video integrity verification summary

Device ID	Original		WhatsApp			YouTube			FFmpeg			Exiftool		
	Min	Max	Min	Max	AUC	min	max	AUC	Min	Max	AUC	Min	Max	AUC
D01	0.005	**0.005**	**0.995**	0.995	1.000	**0.986**	0.986	1.000	**0.985**	0.985	1.000	**0.010**	0.015	1.000
D02	0.016	**0.016**	**0.818**	0.822	1.000	**0.792**	0.801	1.000	**0.662**	0.672	1.000	**0.009**	0.019	**0.627**
D03	0.034	**0.034**	**0.985**	0.985	1.000	**0.991**	0.991	1.000	**0.991**	0.991	1.000	**0.009**	0.043	0.906
D04	0.000	**0.000**	**0.995**	0.995	1.000	**0.986**	0.986	1.000	**0.986**	0.986	1.000	**0.000**	0.010	0.974
D05	0.328	**0.328**	**0.845**	0.849	1.000	**0.825**	0.833	1.000	**0.716**	0.723	1.000	**0.320**	0.330	**0.721**
D06	0.011	**0.011**	**0.814**	0.818	1.000	**0.790**	0.799	1.000	**0.664**	0.668	1.000	**0.002**	0.013	0.822
D08	0.005	**0.005**	**0.995**	0.995	1.000	**0.986**	0.986	1.000	**0.986**	0.986	1.000	**0.010**	0.015	1.000
D09	0.012	**0.012**	**0.818**	0.818	1.000	**0.791**	0.799	1.000	**0.662**	0.668	1.000	**0.002**	0.016	**0.723**
D10	0.011	**0.011**	**0.814**	0.814	1.000	**0.790**	0.797	1.000	**0.664**	0.668	1.000	**0.009**	0.013	0.884
D11	0.000	**0.000**	**0.995**	0.995	1.000	**0.986**	0.986	1.000	**0.986**	0.986	1.000	**0.010**	0.010	1.000
D13	0.011	**0.011**	**0.818**	0.818	1.000	**0.791**	0.797	1.000	**0.664**	0.668	1.000	**0.009**	0.012	0.891
D14	0.007	**0.007**	**0.818**	0.818	1.000	**0.791**	0.795	1.000	**0.664**	0.664	1.000	**0.009**	0.009	1.000
D15	0.326	**0.326**	**0.845**	0.845	1.000	**0.825**	0.833	1.000	**0.716**	0.720	1.000	**0.325**	0.327	**0.768**
D16	0.005	**0.005**	**0.985**	0.985	1.000	**0.991**	0.991	1.000	**0.991**	0.991	1.000	**0.009**	0.014	1.000
D18	0.011	**0.011**	**0.814**	0.814	1.000	**0.790**	0.797	1.000	**0.664**	0.668	1.000	**0.009**	0.013	0.870
D19	0.328	**0.328**	**0.820**	0.849	1.000	**0.799**	0.837	1.000	**0.664**	0.725	1.000	**0.127**	0.427	**0.699**
D20	0.013	**0.013**	**0.814**	0.814	1.000	**0.790**	0.799	1.000	**0.664**	0.668	1.000	**0.006**	0.017	**0.593**
D21	0.000	**0.000**	**0.985**	0.985	1.000	**0.990**	0.990	1.000	**0.990**	0.990	1.000	**0.000**	0.010	0.955
D22	0.000	**0.000**	**0.995**	0.995	1.000	**0.986**	0.986	1.000	**0.985**	0.985	1.000	**0.010**	0.010	1.000
D23	0.000	**0.000**	**0.985**	0.985	1.000	**0.991**	0.991	1.000	**0.991**	0.991	1.000	**0.009**	0.009	1.000
D24	0.027	**0.027**	**0.985**	0.985	1.000	**0.991**	0.991	1.000	**0.991**	0.991	1.000	**0.009**	0.062	**0.618**
D25	0.005	**0.005**	**0.985**	0.985	1.000	**0.991**	0.991	1.000	**0.991**	0.991	1.000	**0.009**	0.014	1.000
D26	0.000	**0.000**	**0.995**	0.995	1.000	**0.986**	0.986	1.000	**0.985**	0.985	1.000	**0.010**	0.010	1.000
D27	0.000	**0.000**	**0.985**	0.985	1.000	**0.991**	0.991	1.000	**0.991**	0.991	1.000	**0.009**	0.009	1.000
D28	0.005	**0.005**	**0.985**	0.985	1.000	**0.991**	0.991	1.000	**0.991**	0.991	1.000	**0.009**	0.014	1.000
D29	0.326	**0.326**	**0.845**	0.845	1.000	**0.825**	0.833	1.000	**0.716**	0.720	1.000	**0.320**	0.327	**0.765**
D30	0.005	**0.005**	**0.985**	0.985	1.000	**0.991**	0.991	1.000	**0.991**	0.991	1.000	**0.009**	0.014	1.000
D31	0.000	**0.000**	**0.995**	0.995	1.000	**0.986**	0.986	1.000	**0.986**	0.986	1.000	**0.010**	0.010	1.000
D32	0.026	**0.026**	**0.985**	0.985	1.000	**0.991**	0.991	1.000	**0.991**	0.991	1.000	**0.009**	0.062	0.821
D33	0.005	**0.005**	**0.995**	0.995	1.000	**0.986**	0.986	1.000	**0.986**	0.986	1.000	**0.010**	0.015	1.000
D34	0.015	**0.015**	**0.814**	0.818	1.000	**0.795**	0.803	1.000	**0.664**	0.672	1.000	**0.006**	0.017	**0.696**

$L_C(Y_i)$ are computed, thus achieving statistics for matching cases (videos belonging to brand C) and for mismatching cases (videos belonging to other brands). We use values $L_C(X_i)$ for building a ROC curve, and report its AUC in Table 4.4: we notice that the AUC is perfect on almost all brands with the exception of *Xiaomi*, that achieves an AUC equal to 0.99. These result were caused by some *Wiko* videos that share a very similar container as to the *Xiaomi* one.

As to model identification, we adopt a similar strategy for testing by considering all available videos from models C; for each $X_i \in C$ we consider an $Y_i \in \overline{C}$ randomly chosen. Then, weights (equation 4.5) are built using all videos from class C, except[12] for X_i, and an equal number of videos from same brands but other model classes. The likelihoods $L_C(X_i)$ and $L_C(Y_i)$ are computed, thus achieving statistics for matching cases (videos belonging to model C) and for mismatching cases (videos belonging to other models). We compute the ROC curve and report the AUC in Table 4.5. We notice that the worst results are obtained for the *OnePlus* brand, where the container structure of the videos belonging to the *A3000* and *A3003* models are not sufficient to perform a significant identification.

It is worth noting that *Apple* devices running a firmware older than iOS 9.x obtain an AUC greater than 0.90 while more recent firmwares result in AUC smaller than 0.81. Clearly, the worst results are obtained with the *iPhone 6 plus* that runs the latest firmware and corresponds to the latest device in the *Apple* family. On the other hand, the best performance is obtained with the *Samsung* brand, where unitary AUC is obtained for all models. Also the *Huawei* brand performs almost perfectly for all models, while resulting in an AUC around 0.80 in the recent models, namely the *P9* and *P9 Lite*. Note that, with the proposed technique, the mean computational time required for a video brand/model identification is up to 0.86 seconds[11].

Brand/Model Classification

As shown in Section 4.3.2, the proposed approach can be exploited to classify the brand/model of a video. In the following we deal only with the model

[12]In principle, it would be correct not to consider videos from the same exemplar in class C. However, given the limited number of available videos for some models, we also allowed videos from the same device exemplar here. Since we do not expect video containers to carry any information about the specific exemplar, the impact of this choice should be negligible.

Table 4.4: Brand identification

Brand	AUC
Apple	1.00
Asus	1.00
Huawei	1.00
Lg	1.00
Oneplus	1.00
Samsung	1.00
Wiko	1.00
Xiaomi	0.99

Table 4.5: Model identification

Apple		Huawei		Samsung		Oneplus	
Model	AUC	Model	AUC	Model	AUC	Model	AUC
iPad 2	0.92	Ascend	1.00	Galaxy S3	1.00	A3000	**0.52**
iPad mini	1.00	Honor 5C	0.95	Galaxy S3 Mini	1.00	A3003	**0.51**
iPhone 4	0.98	P8	1.00	Galaxy S4 Mini	1.00		
iPhone 4S	0.99	P9	**0.79**	Galaxy S5	1.00		
iPhone 5	0.81	P9 Lite	**0.80**	Galaxy Tab 3	1.00		
iPhone 5C	0.79			Galaxy Trend Plus	1.00		
iPhone 6	0.78						
iPhone 6 Plus	**0.53**						

classification task, since it comprises the brand classification too. Let us consider a query video X, whose source device is unknown. The task is to assign to X the most likely source model from a set of known models. To do so, for all available brand classes $\{B_1, \dots B_s\}$, the likelihoods $L_{B_1}(X), \dots L_{B_s}(X)$ are computed and the output brand B^* is chosen as the one achieving the maximum value. Then, given all available model classes for the selected brand, say $\{M_1, \dots M_t \ | M_i \in B^*\}$[13], the likelihoods $L_{M_1}(X), \dots L_{M_t}(X)$ are computed (as defined in the identification pipeline) and the model is chosen as the one achieving the maximum value.

Table 4.6 gives the confusion matrix for the case of Brand classification: we see that the likelihood-metric works perfectly on *Apple*, *Asus*, *LG* and *OnePlus* brands, whereas gain over 80% accuracy on average on the

[13]Here, the notation $M \in B$ means that model M belongs to brand B.

remaining brands. It is worth noting that videos belonging to *Wiko* and *Xiaomi*, while not reaching perfect accuracy, always do appear in the TOP 3 brands according to the likelihood metric. Unfortunately, videos belonging to *Huawei* and *Samsung* are not classified perfectly even in the TOP 3 brands, where the *Huawei* performs better than the *Samsung*.

In the Model Classification tests, we perform the analysis only for those brands of the dataset for which we have more than one model, that is *Samsung*, *Apple* and *Huawei*. The *Samsung* model classification confusion matrix is represented in Table 4.7. The likelihood metric results in a perfect match on almost all models with exception to the *Galaxy S3*, where indeed no video is considered to be belonging to the *Samsung* brand. This is probably due to the D11 firmware, that is known to be an *Anonymous Android*, as the D21 device; indeed, we see in Table 4.6 that 10.81% of *Samsung* videos are assigned to the *Wiko* brand.

The *Huawei* model classification confusion matrix is given in Table 4.8, showing performances in line with the Brand classification case: some *Huawei* videos are not matched to this brand, in fact 10.53% of videos belonging to the *Ascend* and the 63.16% of videos belonging to the *P8* model, are assigned to the *other* class. The TOP 3 model match is perfect on the *Honor 5c*, *P9* and *P9 Lite*, but have the worst results on the *P8* due to the majority of the videos being matched to another brand. It is important to notice that the models *P9* and *P9 Lite* are mismatched, all videos from the *P9 Lite* are matched to the *P9* and viceversa. These results would likely improve by adding to the dataset more devices belonging to the same model.

The *Apple* models confusion matrix is represented in Table 4.9. We obtain always a perfect match on the TOP 3 model, meaning that for each video the exact model match is always in the TOP 3 of the model likelihood. As to the best single match, we obtain the worst performances on the *iPhone 5*, *iPhone 5c*, *iPhone 6 Plus* and *iPad 2*. These unbalanced performances are probably due to the devices firmware version since, as Table 4.10 shows, we do not have for the same model also the same firmware. This means that if the container structure changes between firmwares, the same model cannot be perfectly described with our weights, and instead of detecting the model we obtain a firmware classification, as can be seen on the two *iPhone 6* devices. The D06 and D15 have two possible firmwares, and in indeed the performances on Table 4.9 show that 42% of the tested videos are considered as *iPad mini* that run iOS 8.4 whereas 51% are considered to belong to an

iPhone 6 Plus that runs iOS 10.2. The model classification on the *OnePlus* devices show that all *OnePlus* videos are matched to *A3000* model. Note that, with the proposed technique, the mean computational time required for a video brand/model classification is 8.74 seconds[11].

Table 4.6: Brand classification confusion matrix

cnf%	Apple	Asus	Huawei	Lg	Oneplus	Samsung	Wiko	Xiaomi	TOP3
Apple	100.00	0	0	0	0	0	0	0	100.00
Asus	0	100.00	0	0	0	0	0	0	100.00
Huawei	0	0	**85.26**	2.11	0	0	5.26	7.37	**98.95**
Lg	0	0	0	100.00	0	0	0	0	100.00
Oneplus	0	0	0	0	100.00	0	0	0	100.00
Samsung	0	0	0	0	2.03	**87.16**	10.81	0	**87.16**
Wiko	0	0	0	0	0	0	**81.82**	18.18	100.00
Xiaomi	0	0	0	0	0	0	21.05	**78.95**	100.00

Table 4.7: Model classification confusion matrix for *Samsung* devices

cnf %	Galaxy S3	Galaxy S3 mini	Galaxy S4 mini	Galaxy S5	Galaxy Tab 3	Galaxy Trend Plus	Other	TOP3
Galaxy S3	**0**	0	0	0	0	0	100.00	**0**
Galaxy S3 mini	0	100.00	0	0	0	0	0	100.00
Galaxy S4mini	0	0	100.00	0	0	0	0	100.00
Galaxy S5	0	0	0	100.00	0	0	0	100.00
Galaxy Tab 3	0	0	0	0	100.00	0	0	100.00
Galaxy Trend Plus	0	0	0	0	0	100.00	0	100.00

Table 4.8: Model classification confusion matrix for *Huawei* devices

cnf %	Ascend	Honor 5C	P8	P9	P9 Lite	Other	TOP3
Ascend	**89.47**	0	0	0	0	10.53	**89.47**
Honor5c	0	100.00	0	0	0	0	100.00
P8	0	0	**36.84**	0	0	63.16	**36.84**
P9	0	5.26	0	**0**	94.74	0	100.00
P9 Lite	0	0	0	100.00	**0**	0	100.00

4.5 Remarks

This Chapter proposes an approach for the forensic analysis of video file containers. The core idea is to exploit the differences in the file container structure and content introduced by different manufacturers and models. In particular, we proposed the first formal approach in the state of the art to perform integrity verification, brand/model identification and classification

Table 4.9: Model classification confusion matrix for *Apple* devices

cnf %	**iPad 2**	**iPad Mini**	**iPhone 4**	**iPhone 4S**	**iPhone 5**	**iPhone 5C**	**iPhone 6**	**iPhone 6 Plus**	**TOP3**
iPad 2	**31.25**	0	68.75	0	0	0	0	0	100.00
iPad Mini	0	100.00	0	0	0	0	0	0	100.00
iPhone 4	0	0	100.00	0	0	0	0	0	100.00
iPhone 4S	0	0	3.57	**96.43**	0	0	0	0	100.00
iPhone 5	0	62.75	0	0	**0**	0	0	37.25	100.00
iPhone 5C	0	23.53	0	37.25	1.96	**0**	0	37.25	100.00
iPhone 6	0	42.86	0	0	0	0	**5.71**	51.43	100.00
iPhone 6 Plus	0	0	0	5.26	0	0	0	**94.74**	100.00

based on such features. Extensive experiments were carried on a publicly available dataset, showing excellent results for the integrity verification task, and encouraging results for the brand/model identification and classification tasks. Noticeably, the proposed technique allows to automatically detect manipulations that are performed without video re-encoding, which is an unexplored field in video forensic state of the art.

As a future work, we propose to refine the atom matching strategy presented in Section 4.3 in such a way that atom values that are expected to change between different videos of the same device (e.g., date, video duration, etc.) are compared differently than values which re expected to remain constant. Similarly, the robustness of the method to intra-class variability can be increased by defining refined container comparison strategies.

Table 4.10: VISION dataset main features.

Brand	Model	ID	Software/Firmware	#vOrigin
Apple	iPad 2	D13	iOS 7.1.1	16
Apple	iPad mini	D20	iOS 8.4	16
Apple	iPhone 4	D09	iOS 7.1.2	19
Apple	iPhone 4S	D02	iOS 7.1.2	13
Apple	iPhone 4S	D10	iOS 8.4.1	15
Apple	iPhone 5	D29	iOS 9.3.3	19
Apple	iPhone 5	D34	iOS 8.3	32
Apple	iPhone 5c	D05	iOS 10.2.1	19
Apple	iPhone 5c	D14	iOS 7.0.3	19
Apple	iPhone 5c	D18	iOS 8.4.1	13
Apple	iPhone 6	D06	iOS 8.4	17
Apple	iPhone 6	D15	iOS 10.1.1	18
Apple	iPhone 6 Plus	D19	iOS 10.2.1	19
Asus	Zenfone 2 Laser	D23*	-	19
Huawei	Ascend G6-U10	D33	-	19
Huawei	Honor 5C NEM-L51	D30	Android 6.0/NEM-L51C432B120	19
Huawei	P8 GRA-L09	D28	Android 6.0/GRA-L09C55B330	19
Huawei	P9 EVA-L09	D03	Android 6.0/EVA-L09C55B190	19
Huawei	P9 Lite VNS-L31	D16	Android 6.0/VNS-L31C02B125	19
Lenovo	Lenovo P70-A	D07	-	19
LG electronics	D290	D04	-	19
Microsoft	Lumia 640 LTE	D17	Windows Phone	10
OnePlus	A3000	D25	Android 7.0/NRD90M 15 dev-keys	19
OnePlus	A3003	D32	Android 7.0/NRD90M 138 dev-keys,	19
-	-	-	NRD90M 18 dev-keys	-
Samsung	Galaxy S III Mini GT-I8190	D26	I8190XXAMG4	16
Samsung	Galaxy S III Mini GT-I8190N	D01	I8190NXXAML1, I8190NXXALL6	22
Samsung	Galaxy S3 GT-I9300	D11	-	19
Samsung	Galaxy S4 Mini GT-I9195	D31	I9195XXUCNK1	19
Samsung	Galaxy S5 SM-G900F	D27	Android 6.0.1/G900FXXS1CQAA	19
Samsung	Galaxy Tab 3 GT-P5210	D08	P5210XXUBNK2	37
Samsung	Galaxy Tab A SM-T555	D35	T555XXU1AOE9	16
Samsung	Galaxy Trend Plus GT-S7580	D22	S7580XXUBOA1	16
Sony	Xperia Z1 Compact D5503	D12	14.5.A.0.270_6_f100000f	19
Wiko	Ridge 4G	D21	-	11
Xiaomi	Redmi Note 3	D24	Android 6.0.1/MMB29M	19
-	-	-	V8.1.1.0.MHOMIDI release-keys	-

Chapter 5

High-Level features analysis

In this Chapter we introduce an extended version of the Variation of Prediction Footprint, arguing a justification based on Rate-Distortion curves, and by including B-frames in the second compression stage. The new algorithm will be tested to evaluate double compression identification and first GOP estimation using multiple encoding codecs, constant bit rate and variable bit rate parameters.

5.1 Introduction

In the last decades, the huge technological advancement in the communication and information fields have allowed the burst of digital videos to the point of becoming one of the preferred means to share information. Given their digital nature, these data also convey several information related to their life cycle thus carrying integrity issues that must be studied. The Forensics community, as seen in Section 2.2.3, have been studying the integrity of such contents by taking into account high-level features that are derived from the encoding process. Although many contributions include the macroblocks prediction types as a mean to double compression detection, very few have directly faced the issue of including all type of frames in their analysis. We believe that this gap must be filled, since in new compression algorithms inter-frames play an important role on both the size and the quality of a video content.

In this Chapter we propose a novel extension on the Variation of Pre-

diction Footprint that, in collaboration with the authors in [65], will focus firstly on providing a unique coding-based justification of the VPF effect. It refers to the traces left by the intra-predicted macroblocks decrement concurrently to the increment of skip-predicted macroblocks in P-frames that were encoded as an I-frame in the first compression. In addition, we will focus on the insertion of bi-predicted frames in the coding chain, and discuss when and how the VPF effects shows up in double compressed videos with B-frames. The proposed algorithm will refine the VPF signal by means of a pair of SVM classifiers trained on B-frames and locate those that were encoded as I-frames in the first compression stage. Finally, the VPF-ext signal will be used to perform double compression detection and first GOP identification, in double compressed scenario that include coding algorithm such as MPEG-2, MPEG-4 and H.264/AVC, but also different compression parameters that include Constant Bit Rate and Variable Bit Rate.

The rest of the Chapter is organized as follows: Section 5.2 explains the VPF effect in case of single and double inter-frame types based on rate distortion curves; Section 5.3 formalizes the mathematical framework of the new Variation of Prediction Footprint and describes how can be used to face MF integrity verification; Section 5.4 is dedicated to the experimental validation of the proposed technique along with some state-of-art comparisons; finally, Section 5.5 draws some final remarks and outlines future works.

5.2 The VPF effect

5.2.1 The notation

Figure 5.1 shows the first and second compression workflow of the n-th frame of a raw video $\mathbf{X}$ in a hybrid video coding algorithm. In the first compression the prediction algorithm will define the best type of frame to encode $\mathbf{X}_n$ with, distinguishing between an Intra frame, I-frame, or an Inter-frame, P-frame or B-frame. In addition depending on the frame type, the prediction $\hat{\mathbf{X}}_n$ may contain different types of macroblocks, namely I, P^F, P^B, P^D and S, that respectively correspond to intra predicted, forward predicted, backward predicted, bidirectional predicted and skip predicted macroblocks.

Let us define the number of macroblocks, of a specific prediction type, belonging to the **n**-th frame as I_n, P_n^F, P_n^B, P_n^D and S_n.

The same notation will be used in a double compression scenario. In

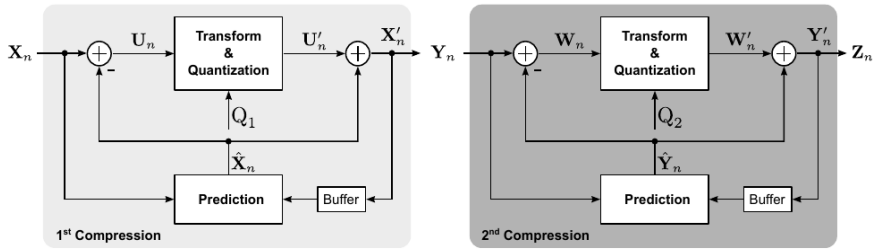


Figure 5.1: Double compression scheme.

particular let us denote with $\mathbf{Y}_n$ and $\mathbf{Z}_n$ the n-th frame of a single and double encoded video, respectively. In order to facilitate the reader, we use superscripts {I, P, B} to make explicit the kind of frame: for example, by writing $\mathbf{Y}_n^{\mathrm{I}}$ we mean that the n-th frame of the single compressed video is an intra-coded frame, while $\mathbf{Z}_n^{\mathrm{B}}$ denotes that the n-th frame of the double compressed video is a bidirectionally predicted frame.

Let us consider the double compression depicted in Figure 5.4 we will refer to the group of consecutive B-frames as the *B-frame sub-GOP*, or just sub-GOP. We will refer with $\mathbf{Z}_{1,P}^{B}$ to the first B-frame in the sub-GOP, namely $\mathbf{Z}_2^B$ corresponding in the first compression to $\mathbf{Y}_2^P$.

5.2.2 Predictions via JDR

The main goal of an encoder is to efficiently represent an input sequence of frames using the available *elements* of a particular video coding standard. To achieve this, the encoder minimizes under certain constraints the distortion between the original sequence $\mathbf{X}_n$ and its reconstruction after encoding $\mathbf{X}_n^{'}$. This minimization problem can be solved using the Lagrangian optimization [60] resulting in the following

$$\mathcal{J}(X_n, X_n^{'}) = \mathcal{D}(X_n, X_n^{'}) + \lambda \mathcal{R}(X_n^{'}), \tag{5.1}$$

$$\text{minimize} \quad \mathcal{J}(X_n, X_n^{'}) \tag{5.2}$$

where $\mathcal{J}$ and λ are respectively the Lagrangian functional and multiplier; $\mathcal{D}(X_n, X_n^{'})$ represents a function that measures the distortion between the

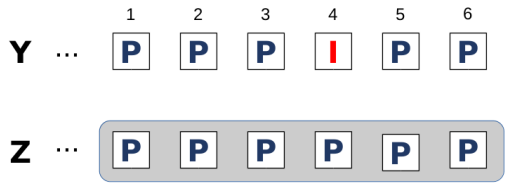


Figure 5.2: Double encoding in presence of P-frames in the second compression.

original frame and the reconstructed one, and $\mathcal{R}(X_n^{'})$ provides the number of bits needed to encode the predicted frame $X_n^{'}$. When this strategy is applied for the macroblock type decision, the minimization of the Lagrangian functional for each MB-type, $\mathcal{J}_{mb}(X_n, X_n^{'})$, yields the following minimization problem

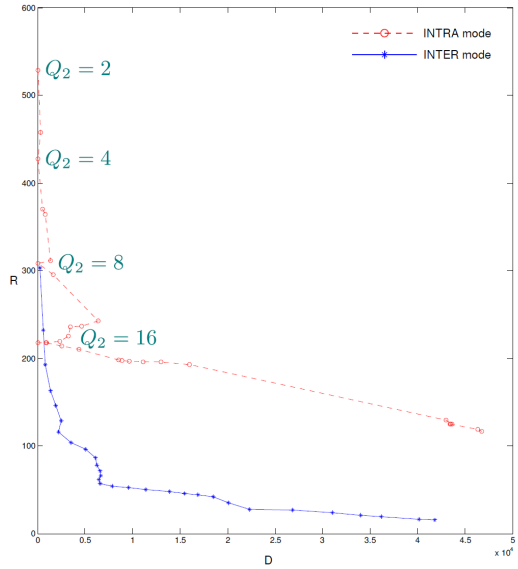
$$\min_{mb} \left(\mathcal{D}(X_n, X_n^{'}) + \lambda_{mb} \mathcal{R}(X_n^{'}) \right) \tag{5.3}$$

where $mb \in \{I, P^F, P^B, P^D, S\}$ and λ_{mb} is the Lagrange multiplier for the selected type whose value is obtained as a function of the quantization parameter $Q_1 \in \mathbb{R}$. Finally a macroblock type is chosen by the encoder if the corresponding value on Equation 5.3 is the smallest one, in the following we will discuss how this mathematical description explains the VPF effect on double encoded videos.

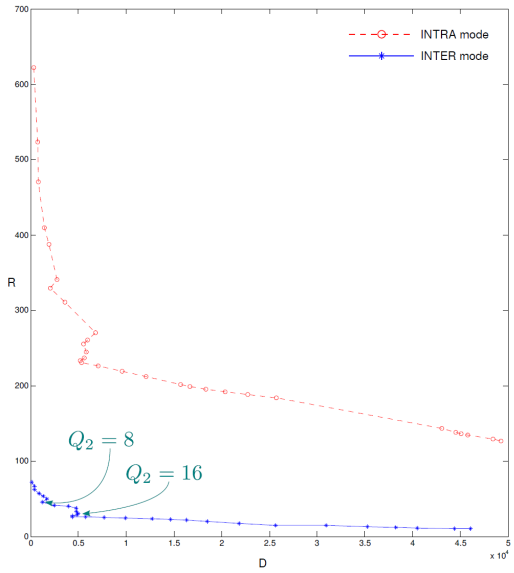
5.2.3 Double Compression

P Inter-frames Figure 5.3 depicts the Rate-Distortion curves for a double encoding scenario with I-frames and P-frames, as in Figure 5.2, by means of the MPEG-2 encoding algorithm for both compressions, with $Q_1 = 16$, $G_1 = 10$, $Q2 \in [2, 31]$ and $G_2 = 33$.

If we take under consideration how frame-4 and frame-6 are encoded in $\mathbf{Z}$, the encoder will evaluate the R-D curves in Figure 5.3(a) to predict $\mathbf{Z}_4^P$, that corresponds to an I-frame in the first encoding, namely $\mathbf{Y}_4^I$; whereas will evaluate the R-D curves in Figure 5.3(b) to predict $\mathbf{Z}_6^P$, that corresponds to a P-frame in the first encoding, namely $\mathbf{Y}_6^P$. It should be clear that the VPF effect can be derived from these curves. Let us consider $\mathbf{Z}_{i,P}^P$ and the R-D curves in Figure 5.3(b), in this case the VPF effect does not manifest,



(a) I-frame *falls* over P-frame



(b) P-frame *falls* over P-frame

Figure 5.3: Rate-Distortion curves for inter (blue) and intra (red) prediction modes averaged across all macroblocks in a double compression scenario with $Q_1 = 16$.

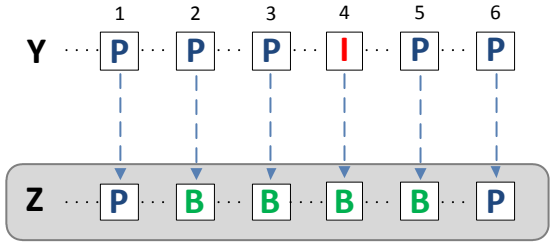


Figure 5.4: Double encoding scenarios in presence of B-frames in the second compression.

indeed from the encoder point of view is much more convenient to predict the $\mathbf{Z}_i$ frame with P^F and S-MBs, with respect to including intra predicted macroblocks. Much more interesting is the representation in Figure 5.3(a) where in some cases the intra-curve is much more convenient that the inter-curve, although when the quantization parameter in the second compression is much stronger, the VPF effect does not show up, and the normal behaviour of the encoder take place, thus preferring inter-prediction to intra-prediction.

B Inter-frames The addition of B-frames allows to increase the compression ratio at the cost of a higher coding complexity, so it is commonly adopted when the encoding device has enough computational resources available, and real-time processing is not requested. Common video capturing devices, like camcorders or smartphones, do not usually meet these requirements, and limit themselves to the use of P-frames. On the contrary, video editing tools usually run on powerful devices and are not subject to strict time constraints, so they will probably make use of B-frames to optimize the encoding.

In the following, we extend our analysis to account for the presence of B-frames in one or both the encodings. Actually, we expect the use of B-frames in the first encoding not to have a sensible impact on the proposed method, since the position of I-frames depends on the size of the GOP and not on its structure. On the other hand, we expect the presence of B-frames in the second encoding to affect the VPF, because it enlarges the set of possible choices for encoding each MB. Let us consider the scenario in Figure 5.4, where the frame $\mathbf{Y}_4^{\mathrm{I}}$ is re-encoded in the B-frame $\mathbf{Z}_4^{\mathrm{B}}$. It is worth recalling that, due to the presence of B-frames, the second compression encoding order is as follows:

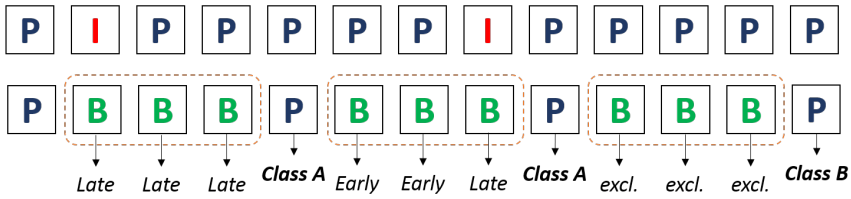
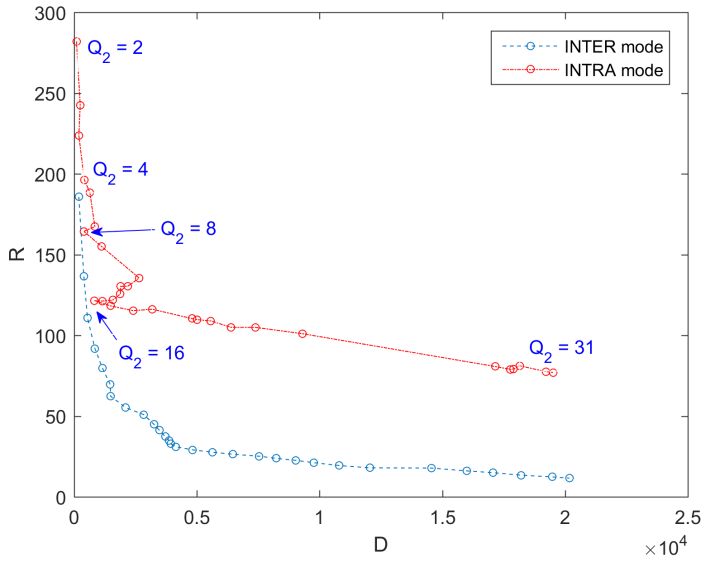


Figure 5.5: This figure graphically shows the criterion for selecting *Class A* and *Class B* P-frames, and *Early* and *Late* B-frames.

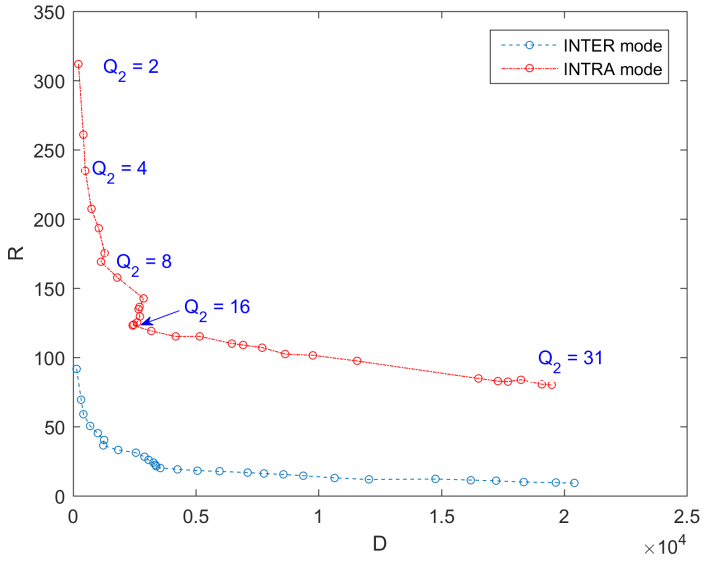
- $\mathbf{Z}_1^{\mathrm{P}}$ is encoded;
- $\mathbf{Z}_6^{\mathrm{P}}$ is encoded;
- $\mathbf{Z}_{2,3,4,5}^{\mathrm{B}}$ are encoded, starting from $\mathbf{Z}_2^{\mathrm{B}}$.

Thus, $\mathbf{Z}_6^{\mathrm{P}}$ prediction is computed from $\mathbf{Z}_1^{\mathrm{P}}$. In this situation, the presence of an I-frame in the previous compression at frame 4 plays an important role: predicted MBs in $\mathbf{Z}_6^{\mathrm{P}}$ reference to $\mathbf{Z}_1^{\mathrm{P}}$, that is the re-encoded version of $\mathbf{Y}_1^{\mathrm{P}}$. Since $\mathbf{Y}_1^{\mathrm{P}}$ and $\mathbf{Y}_6^{\mathrm{P}}$ belong to different GOPs in $\mathbf{Y}$, we expect a significant residual error in $\mathbf{Z}_6^{\mathrm{P}}$, that should imply a reduced number of S-MBs.

This conjecture can be easily verified by adapting the experiment reported in Figure 5.3. To do so, we divide the P-frames of a double compressed video in two sets: *Class A* P-frames, i.e., those whose preceding sub-GOP re-encodes an I-frame, and *Class B* P-frames, i.e., those whose preceding sub-GOP does not re-encode any I-frame. Figure 5.5 provides a graphical explanation. Then, we plot in Figure 5.6 the rate-distortion curves for INTRA and INTER mode averaged on *Class A* P-frames, Figure 5.6(a), and on *Class B* P-frames, Fig. 5.6(b). We notice that the INTER and INTRA mode curves are significantly closer in the former case, especially for $Q_2 = 8$ and $Q_2 = 16$, as it was also in Figure 5.3(a) compared to Figure 5.3(b). Therefore, we can state that the presence of an I-frame in the first encoding between two P-frames is likely to cause a decrease of skipped and predicted MBs in the P-frame that follows, and an increase of I-MBs. In other words, the presence of B-frames will not cancel the VPF, but simply shift it to the next P-frame. In the following we show how this shift can be measured and compensated, thus restoring the original periodicity of the signal.



(a) *Class A* P-frames



(b) *Class B* P-frames

Figure 5.6: Rate-Distortion curves for inter (blue) and intra (red) prediction modes averaged across all macroblocks of all *Class A* and *Class B* P-frames of the *News* sequence.

B-frames differ from P-frames in that they allow referencing each MB to one of a previous I/P reference frame (forward-prediction), to one of a following I/P reference frame (backward prediction), or to combine information from both past and future I/P reference frames (bidirectional-prediction). The choice on whether using a forward-predicted MB, backward-predicted MB, or bidirectional-predicted MB is once more guided by the rate-distortion principle.

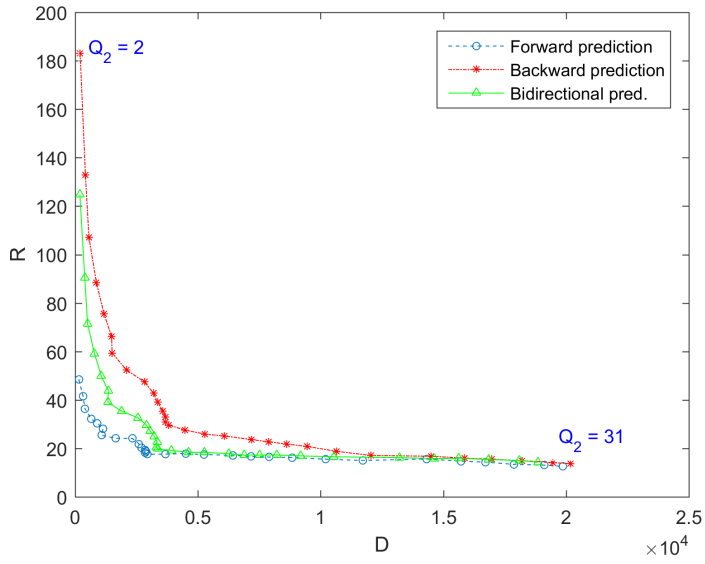
In a single, almost static, compressed sequence, the encoder uses backward prediction more likely when encoding B-frames near to the last reference frame, and gradually turns toward forward prediction as the future reference frame approaches. The intuition behind this fact is straightforward: frames that are near in time are normally more similar, and the prediction is more convenient.

Let us now repeat the same reasoning for the double encoding case, focusing on the scenario depicted in Figure 5.4 to derive the idea. The frame $\mathbf{Z}_4^B$ must encode $\mathbf{Y}_4^I$ by referencing either $\mathbf{Z}_1^P$ or $\mathbf{Z}_6^P$ (which are, respectively, the decoded version of $\mathbf{Y}_1^P$ and $\mathbf{Y}_6^P$). Since $\mathbf{Y}_6^P$ belongs to the GOP initiated by $\mathbf{Y}_4^I$, the similarity of its encoded version $\mathbf{Z}_6^P$ with $\mathbf{Y}_4^I$ is likely to be much higher than the one of $\mathbf{Z}_1^P$. As a consequence, the frame $\mathbf{Z}_4^B$, and also the following ones, are more conveniently encoded using $\mathrm{P^B}$-MBs, even when the preceding P-frame is nearer in time.

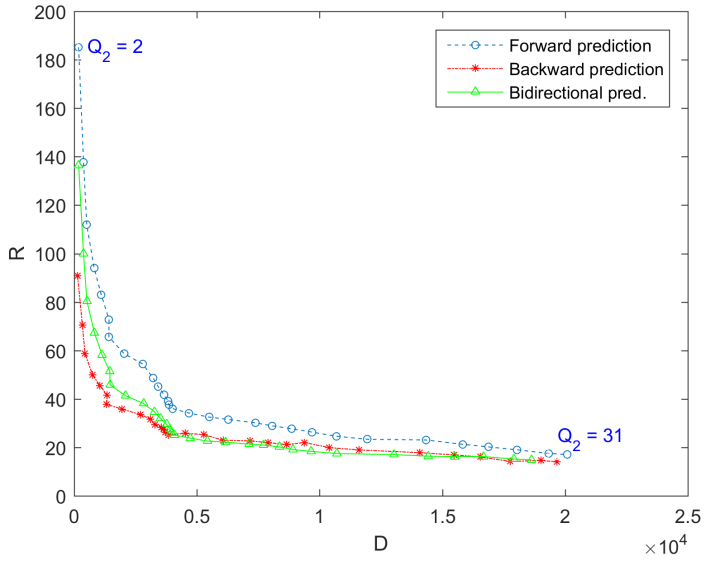
This conjecture can be easily verified resorting once again to the analysis of rate-distortion curves. Let us divide the B-frames of the double encoded sequence in two classes (graphically explained in Fig. 5.5): *Early* B-frames are those having an I-frame in the previously encoded sequence "falling" over them or over a successive B-frame of the same sub-GOP, and *Late* B-frames are those having an I-frame in the previously encoded sequence falling over a preceding B-frame of the same sub-GOP. B-frames that are neither *Early* nor *Late*, meaning their sub-GOP does not re-encode any I-frame, are excluded from the analysis because they do not carry the VPF effect. We compute the rate-distortion curves for the forward, backward and bidirectional prediction modes under the same settings used for previous experiments [65], setting the GOP length of the first encoding to 12 and that of the second encoding to 34, and allowing the use of 3 consecutive B-frames in the second encoding. Fig. 5.7(a) shows the curve for *Early* B-frames, and Fig. 5.7(b) shows that for *Late* B-frames. As expected, we see that for *Early* B-frames using the forward prediction is much more convenient, while the opposite holds for

Late B-frames.

Therefore, we expect to observe an abrupt change in the use of P^F-MBs and P^B-MBs in those sub-gops of the double compressed sequence that re-encode an I-frame of the single compressed sequence. This is indeed the case, as Fig. 5.8 shows: while in sub-gops that do not re-encode any I-frame we see a progressive increase in the number of P^B-MBs compared to P^F-MBs, that is because the new information is likely to appear in the future, on the other hand we notice an abrupt increase in sub-gops that re-encode a previous I-frame. In particular, the strongest variation is observed exactly in the B-frame that re-encodes the previous I-frame. This fact suggests that by analysing the prediction type within B-frames MBs we can understand where the previous encoding I-frame occurred within a sub-GOP, thus allowing us to measure the previously mentioned VPF shift and compensate it.



(a) *Early* B-frames



(b) *Late* B-frames

Figure 5.7: Rate-Distortion curves for forward (blue), backward (red), and bidirectional (green) prediction averaged across all bidirectional macroblocks of all *Early* and *Late* B-frames of the *News* sequence.

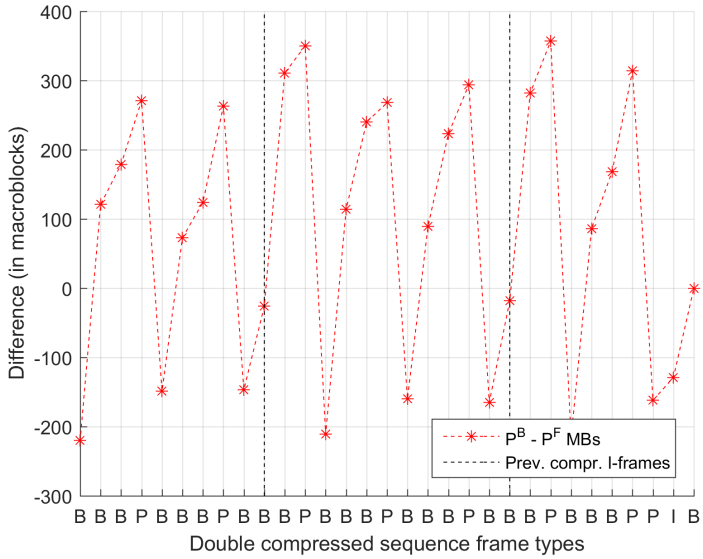


Figure 5.8: Difference in the number of P^B-MBs and P^F-MBs for the double encoded *News* sequence (Q1 = 16, Q2 = 12). Only frames in the range [39-69] are plotted for better visibility.

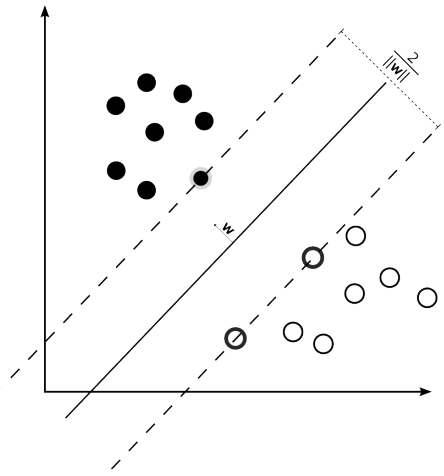


Figure 5.9: A simple example of a linear SVM classification.

5.3 VPF extension

In the following we will describe the VPF extension algorithm, firstly we will discuss the basics of the SVM algorithm, the new prediction framework will be described and we will conclude this section with the technique used to perform multimedia forensics analysis on the integrity verification problem.

5.3.1 SVM basics

The scientific community often deals with *data* that are linked with some sort of relationship. In many cases, unfortunately, handling huge data contents and understanding the overall behaviour is not always feasible. To solve this problems many researchers have investigated automatic machine learning solutions to perform classification.

One of the most used solutions is the *supervised learning* algorithm known as Support Vector Machines [23] represented using a toy example in Figure 5.9. In the case of a two-class pattern problem, as in Figure 5.9, where the classes are linearly separable the SVM selects from among the infinite number of linear decision boundaries the one that maximizes the *margin* between the two classes, where the *margin* is defined as the sum of the distances to the hyperplane from the closest points of the two classes. The data

points that are closest to the hyperplane are used to measure the *margin*; hence called *support vectors*. If the two classes are not linearly separable, the SVM tries to find the hyperplane that splits the examples as cleanly as possible by means of a *soft margin* solution, that introduces a set of *slack* variables that allow for a certain degree of misclassification.

SVM can also be extended to handle non-linear decision surfaces by applying the *kernel trick* [17], that corresponds to projecting the input data onto a high-dimensional feature space $\mathcal{V}$, where the classes are easily separable, using *kernel functions* and formulating a linear classification problem in that feature space.

Supposing that our examples lie in a f-dimensional feature space $X \subseteq \mathbb{R}^f$, and considering a feature map $\phi : X \to \mathcal{V}$ and the inner product in the $\mathcal{V}$ space $\langle \cdot, \cdot \rangle_{\mathcal{V}}$, the kernel function k can be written as

$$\mathsf{k}(x_i, x_j) = \langle \phi(x_i), \phi(x_j) \rangle_{\mathcal{V}}. \tag{5.4}$$

Commonly used kernel functions are:

- **Linear**: $\mathsf{k}(x_i, x_j) = \langle x_i, x_j \rangle$,
- **RBF** - Radial Basis Function: $\mathsf{k}(x_i, x_j) = exp(-\gamma \|x_i - x_j\|^2) \quad \gamma \in \mathbb{R}$.

SVM was initially designed for binary problems, when dealing with multiple classes, an appropriate multi-class method is needed. *Knerr et al.* proposed in [47] the *one versus one* approach, that perform pair-wise comparisons between m-classes. Thus, all possible two-class classifiers are evaluated, each classifier being trained on only two out of m-classes. Applying each classifier to the test data vectors gives one vote to the winning class, then the data is assigned to the label of the class with most votes.

5.3.2 Prediction framework

We evaluated the contribution to the DC identification given by only B-frames and in particular by the P^F,P^B and P^D macroblocks quantities. In Section 5.2 we noticed that forward and backward macroblocks change their behaviour in occasion of an I-frame encoded as a B-frame, indeed Figure 5.10 shows the frame differences, $d = P^F - P^B$, for *Akiyo* double encoded in H.264 with constant bit rate. In Figure 5.10(a) we compared such differences for normal B-frames, $\mathbf{Z}^B_{i,P}$, and B-frames previously encoded as I-frames, $\mathbf{Z}^B_{i,I}$, it is clear that these frames can be separated nicely, in fact most of the $\mathbf{Z}^B_{i,I}$

lie in the range $[0.06, -0.07]$ and $\mathbf{Z}^B_{i,I} \in \{d1 | d > 0.06 \cup d \leq -0.08\}$. Although this information in quite interesting, it is not sufficient for us to determine whether a sub-GOP re-encodes an I-frame; we also need to understand which B-frame of the subgop is the re-encoded version of the I-frame. In particular the sub-GOP must be taken into account, in particular with $\mathbf{Z}^B_{i,I}$ we want to address B-frames in the i-th position of the sub-GOP that in the first compression were encoded as I-frames. Therefore in Figure 5.10(b), the same data is depicted, in red we have $\mathbf{Z}^B_{1,I}$, in magenta $\mathbf{Z}^B_{2,I}$ and in blue $\mathbf{Z}^B_{3,I}$, unfortunately these classes of B-frames cannot be easily separated, thus a supervised algorithm can be used to identify the $\mathbf{Z}^B_{i,I}$ frames.

Feature extraction So far, we noticed that in B-frames the contribution of the new-macroblock types can be exploited to identify B-frames that carry the VPF effect. For each B-frame in a video, the macroblock types cardinality is extracted. Let us call $\mathcal{S}_B$ a set of consecutive B-frames encoded in $\mathbf{Z}$ as $\mathcal{S}_B = \{\mathbf{Z}^B_1, \mathbf{Z}^B_2, \cdots, \mathbf{Z}^B_{n_B}\}$ where $n_B \in \mathbb{N}$ identifies the total number of consecutive frames of type B. With $\mathbf{p}^F \in \mathbb{R}^{n_B}$, $\mathbf{p}^B \in \mathbb{R}^{n_B}$ and $\mathbf{p}^D \in \mathbb{R}^{n_B}$ we refer to the number of forward predicted, backward predicted and bidirectional predicted macroblocks for each frame of the $\mathcal{S}_B$ sub-GOP. Thus the feature matrix that describes the VPF effect on B-frames is derived as

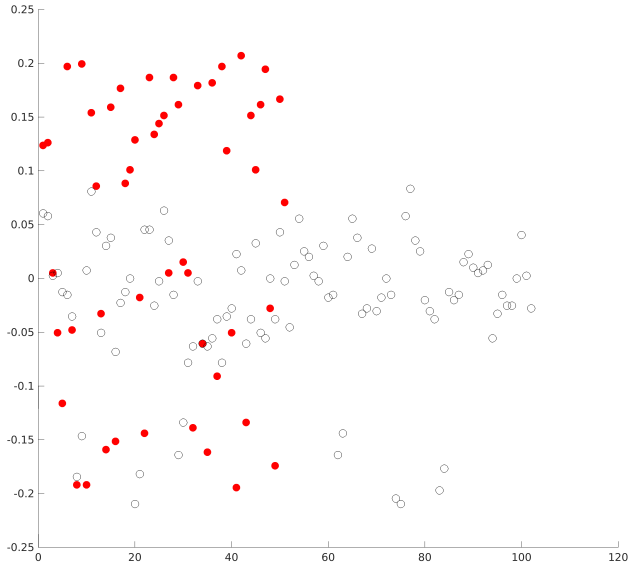
$$\mathbf{F} = \left[\mathbf{p}^F | \mathbf{p}^B | \mathbf{p}^D\right] \in \mathbb{R}^{n_B \times 3} \qquad \forall \mathcal{S}_B \in \mathbf{Z}, \tag{5.5}$$

whereas the label $\mathbf{l}$ vector will be long $K \in \mathbb{R}$ as the total number of B-frames sub-gops in $\mathbf{Z}$ and it will assign to each sub-GOP, 0 if none of the frames were encoded as an I-frame in the first compression, or the B-frame sub-GOP index to identify the position of a fallen I, as in

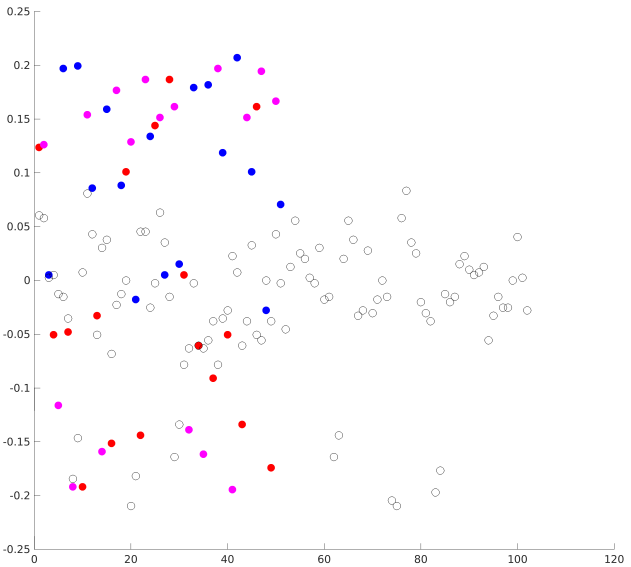
$$\mathbf{l} = [l_1, l_2, \cdots, l_K]^T \;\; l_i \in [0, n_B]. \tag{5.6}$$

SVM framework Given a training set $\mathbb{Z}^{n_B}$ of double compressed[1] videos and n_B consecutive B-frames, we trained in *cross-validation* two SVM classifiers, namely **Clf1** and **Clf+**.

[1]For each compression algorithm a set of CBR and VBR parameters are taken into account to perform the actual encoding, thus leading to SVM classifiers that refer to the triplet Codec-Mode-n_B.



(a) clf1: B-frames previously encoded as I frames in red.



(b) clf2: 1-red,2-magenta,3-blue in sub-GOP

Figure 5.10: $P^F - P^B$ for each B-frame in a H.264 DC scenario with $B_1 = 100k$, $B_2 = 400k$ and sub-GOP size 3.

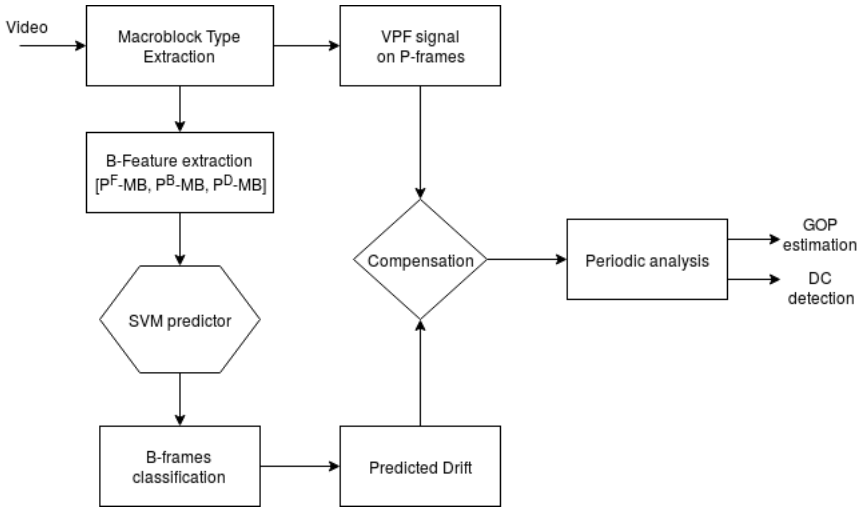


Figure 5.11: VPF-ext framework workflow

Clf1 uses an *RBF kernel* and the feature-label[2] pair $(\mathbf{F}, \mathbf{l})$ to determine whether a B-frame sub-GOP $\mathcal{S}_B$ hides an I-frame in the first compression or not.

Clf+ uses a *linear kernel* and a subset of the pair $(\mathbf{F}, \mathbf{l})$ that contains only $\mathcal{S}_B$ that re-encode a previously encoded I-frame. **Clf+** handles a multi-class problem using the one-vs-one scheme thus predicting the B-frame index in the range $[1, n_B]$ for each $\mathcal{S}_B$.

Therefore, to determine if a set of features extracted from a double encoded video, may contain hints on B-frames encoded as I-frames, the previous trained classifiers will be used in series. **Clf1** will determine if sub-GOP anomalies are present, and on those data **Clf+** will evaluate where the I-frames are located.

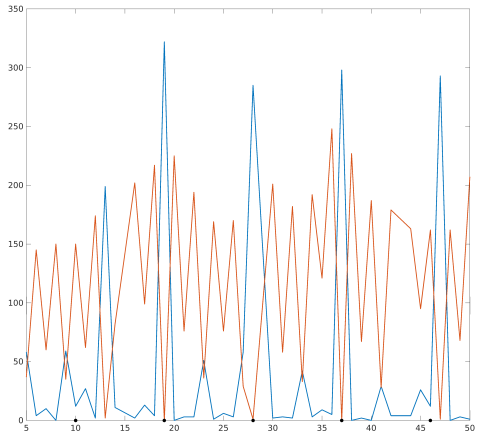


Figure 5.12: I-MBs(red) and S-MBs(blue) comparison limited to frames [5.50] in *Bus* double compressed with H.264. The first compression GOP is represented with black dots.

5.3.3 MF analysis

In the following, the algorithm depicted in Figure 5.11 will be described and it will be shown how to exploit it with Multimedia Forensics problems such as double compression identification and first GOP estimation.

The VPF-ext signal Given a double encoded video **Z** containing M-frames of type I, P and n_B B-frames, the macroblock types are extracted for each frame; Then the P-frame footprint is extracted by taking into account the number of I and S macroblocks for each frame of the set $\mathcal{P}$ affected by the VPF-effect as introduced in [65], the following notation will be used:

$$\mathcal{P} = \{n \in \{0, M-1\} \mid (I_{n-1} < I_n) \wedge (I_n > I_{n+1}) \wedge (S_{n-1} > S_n) \wedge (S_n < S_{n+1})\}, \tag{5.7}$$

[2]The feature matrix will be fed to the classifier after a normalization process based on the maximum number of macroblocks in the video frame.

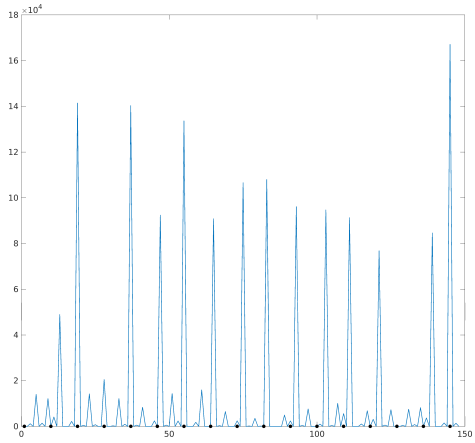


Figure 5.13: VPF signal(blue) computed on *Bus* using first GOP equal to 9(black dots) and a CBR in a H.264 compression algorithm with $n_B = 3$.

$$E_n = |\,(I_n - I_{n-1}) \cdot (S_n - S_{n-1})\,| + |\,(I_{n+1} - I_n) \cdot (S_{n+1} - S_n)\,|, \tag{5.8}$$

$$v_n = \begin{cases} E_n & \text{if } n \in \mathcal{P}, \\ 0 & \text{otherwise.} \end{cases} \tag{5.9}$$

The VPF effect formalized by the v_n signal is shown in Figure 5.13, and the contribution of intra macroblocks and skip macroblocks is shown in Figure 5.12 with respect to a double compressed video with H.264, three consecutive B-frames in constant bit rate with the first compression compressed at 100kb/s and the second one at 700kbit. It should be clear that the VPF effect is present in this double compression scenario that includes B-frame, but as can be seen in Figure 5.12 the frame $\mathbf{Z}^B_{46}$, that was encoded as an I-frame in the first compression, the I-MBs, S-MBs variations are drifted to the nearest P-frame corresponding to frame $\mathbf{Z}^P_{47}$.

The introduction of B-frames in the second compression stage *drifts* the VPF effect into the next P-frame in the sequence, as deeply discussed in Section 5.2. The the novelty introduced with this prediction framework is to

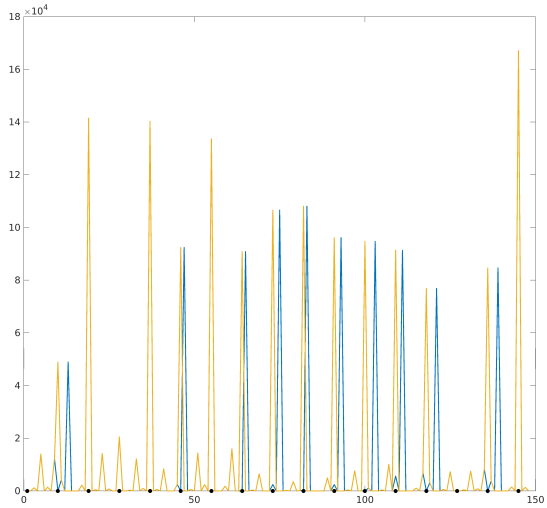


Figure 5.14: VPF-ext signal (orange) compared to the original VPF signal (blue) on H.264 double compressed *Bus*. Notice that in the VPF-ext signal the periodicity is re-established.

identify the B-frames that were previously encoded as an I-frame by means of the SVM classifier pairs. For each S_B in the double compressed video $\mathbf{Z}$, the SVM classification produces the predicted vector $\hat{\mathbf{I}}$ where the location of $\mathbf{Z}^B_{i,I}$ is determined for $i \in [1, n_B]$. Let us denote with $k \in \mathcal{P}$ and $\mathbf{Z}^P_k$ the first P-frame encoded after the sub-GOP $\mathcal{S}^k_B$, then the $\hat{l}_k$[3] prediction will be used to *shift back* the v_k peak to the position $k + \left(\hat{l}_k - n_B - 1\right)$, thus correcting the v-signal periodicity. The drift correction is depicted in Figure 5.14.

Then the periodicity of the signal-v will be studied using the same technique of [65]: firstly if a repeated period is found the first GOP estimation will follow, otherwise the video will be classified as compressed once. The signal-v is evaluated by a two step framework, *candidate GOP selection* and *evaluation*.

The *candidate GOP selection* determines the set of possible G_1 GOPs in the first compression by means of the Greatest Common Divisors (GCD) metric, thus building the set of candidates $\mathcal{C}_G$ as follows:

$$\mathcal{C}_G = \{g \in [2, M] \mid \exists (n_1, n_2) \in \mathcal{P},\ GCD(n_1, n_2) = g\}, \tag{5.10}$$

Then, for each candidate $g \in \mathcal{C}_G$ is applied the *goodness* function $\phi_G : \mathcal{C}_G \to \mathbb{R}$ that determines how well the candidate g describes the signal-v period. Furthermore, the function ϕ_G has to take into account noisy peaks coming from multiples of g, the absence of peaks at multiples of g but also periodic components with a period smaller than g but stronger in amplitude. Thus $\phi_G(g)$ can be derived as ϕ from [65].

DC identification The double compression identification can be determined after the $\phi_G(g)$ computation on each candidate g as

$$DC_{\mathbf{Z}} = \begin{cases} 1, & \text{if } \max_{g \in \mathcal{C}_G} \phi_G(g) > T_\phi, \\ 0, & \text{otherwise,} \end{cases} \tag{5.11}$$

where $T_\phi \in \mathbb{R}$ is an appropriate threshold value, and $DC_{\mathbf{Z}} = 0$ means that the video $\mathbf{Z}$ is classified a single compressed one.

[3]It is important to note that *drift* is corrected only if $\hat{l}_k > 0$, meaning that an I-frame has fallen in the $\mathcal{S}^k_B$ sub-GOP.

GOP estimation In case of $DC_{\mathbf{Z}} = 1$, $\mathbf{Z}$ is identified as a double encoded video, the first GOP estimation can be carried out and performed as

$$\hat{G_1} = \arg\max_{g \in \mathcal{C}_G} \phi_G(g). \tag{5.12}$$

5.4 Performance evaluation

In this Section we show how the proposed method can be exploited to perform double compression identification and first GOP estimation. We will firstly describe the configuration settings used to perform the evaluation of the VPF-ext algorithm. Then we will discuss the performances on the Constant Bit Rate and Variable Bit Rate settings considering both the double compression identification and the GOP estimation. In addition a comparison with a state of the art algorithm will be used for the GOP estimation in a VBR scenario.

5.4.1 Configuration settings

We tested the proposed approach on a dataset of 29 raw sequences[4] with at least 140 frames[5]. Each sequence has undergone a double compression conveyed using *FFmpeg*[6] and `x264`[7] sofware depending on the encoding algorithm. In addition, the first compression uses a *profile* that does not allow the introduction of B-frames, the *baseline profile*, and the second compression, the one we are interested in, is compressed with the *default profile* thus supports a fixed number of consecutive B-frames. The main reasons behind this *constraint* is motivated by the fact that most acquisition devices do not include B-frames in the first encoding since the storing process has to be in real time, whereas the second compression, can vary, and B-frames can be inserted.

Table 5.1 and Table 5.2 report respectively the overall parameters used in the training phase and the testing one. Is is important to notice that in a Variable Bit Rate compression mode, the quantization values used for MPEG-2/4 and H.264/AVC are different, that is because, as in *Bestagini et. al* [15], the Peak to Noise Ratio has been used as a metric to obtain the same video quality between an MPEG-X compression and an AVC one. Furthermore, the two sets of GOP sizes are chosen in such a way that for each number of consecutive B-frames the overall amount of I-frames previously

[4]The videos depicted in Table 5.1 and Table 5.2 have been downloaded in YUV 4:2:0 from `https://media.xiph.org/video/derf/`.

[5]28 videos out of 29 have at least 240 frames, whereas *Bus* is the only sequence with 140 frames.

[6]ffmpeg-3.0.1 was download from `https://ffmpeg.org/`.

[7]x264-snapshot-20160424-2245 was downloaded from `https://www.videolan.org/developers/x264.html`.

encoded are well distributed in the sub-GOP range $[1, n_B]$.

Compression configuration The compression parameters used in the 2nd encoding are represented in the following Listing[8], we provide an example on how to perform CBR, in here with `x264` and *FFmpeg* MPEG-4, and VBR with *FFmpeg* MPEG-2. In addition, we will refer to CBR with `--bitrate` and `-b:v`, and VBR with `-qmin, -qmax` and show how each encoder is set up, the profile will not be made explicit since we will insert the parameter `--bframes` or `-bf` to set the number of consecutive B-frames.

Listing 5.1: An example of double compression settings for VBR and CBR scenarios.

```
x264 -v  --bframes 3 --b-adapt 0 --bitrate 700 --vbv-bufsize 10000
--subme 6 --trellis 0 --no-scenecut --scenecut 0 --aq-mode 0 -I 14
--frames 240 -o outputVideo.h264 --input-res 352x288 inputVideo.yuv

ffmpeg -f rawvideo -pix_fmt yuv420p -s 352x288 -i inputVideo.yuv
-vcodec mpeg2video -g 14 -q:v 1 -qmin 1 -qmax 1 -bf 3  -flags cgop
-sc_threshold 1000000000 -cmp sad -subcmp sad -mbcmp sad -mbd 0
-vframes 240 -y outputVideo.mpeg

ffmpeg -f rawvideo -pix_fmt yuv420p -s 352x288 -i inputVideo.yuv
-vcodec mpeg4 -g 14 -b:v 100k -bf 3 -flags cgop
-sc_threshold 1000000000 -cmp sad -subcmp sad -mbcmp sad -mbd 0
-vframes 240 -y outputVideo.mp4
```

Classification configuration The SVM classifiers introduced in Section 5.3 have been trained and tested by means of the *Sklearn Library*[9] with *Python*[10]. *Sklearn* performs a Python-implementation of the *libsvm* C-library[11].

The double encoded videos obtained from Table 5.1 were grouped with respect to the 2nd encoding algorithm, number of B-frames and compression modes. Then are fed to *Sklearn.SVC* in a five-video cross-validation, consisting of samples of the same video not spread between the training and the validation sets but belong to either one. We want to further address that the

[8] We suggest the reader to read the relative documentation in order to have a better understanding of the parameters used.

[9] Sklearn 0.18.2 can be download from `http://scikit-learn.org/stable/`.

[10] Python 2.7.12 can be downloaded from `https://www.python.it/download/`.

[11] For more information on *libsvm* the reader can refer to `https://www.csie.ntu.edu.tw/~cjlin/libsvm/`.

Table 5.1: Training compression configuration

	1st encoding			**2nd encoding**		
Encoder	MPEG-2	MPEG-4	H.264/AVC	MPEG-2	MPEG-4	H.264/AVC
VBR	{2, 4, 7}		{20, 23, 29}	{1, 5, 10}		{10, 23, 32}
CBR	{100, 300, 500} kb/s			{200, 400, 700} kb/s		
GOP size	{9, 18}			{14, 30}		
n_B frames	{3, 4, 5}					
YUV videos	Akiyo, Bowing, Bridge-close, Bridge-far, Coastguard, Container, Deadline, Flower, Football, Foreman, Hall, Highway, Husky, Mobile, Mother-daughter, News, Pamphlet, Paris, Silent, Tempete					
Video resolution	352×288 pixels					
Frames	240					

Table 5.2: Testing compression configuration

	1st encoding			**2nd encoding**		
Encoder	MPEG-2	MPEG-4	H.264/AVC	MPEG-2	MPEG-4	H.264/AVC
VBR	{2, 4, 7}		{20, 23, 29}	{1, 5, 10}		{10, 23, 32}
CBR	{100, 300, 500} kb/s			{100, 300, 500} kb/s		{200, 400, 700} kb/s
GOP size	{9, 18}			{14, 30}		
n_B frames	{3, 4, 5}					
YUV videos	Bus, City, Crew, Harbour, Ice, SignIrene, Soccer, Students, Waterfall					
Video resolution	352×288 pixels					
Frames	$\{140, 240\}$					

cross-validation has been performed to determine the best pair *C-gamma* by means of a logarithmic scale as in

$$\mathrm{C} \quad = \{\log_2(x) \mid x \in [2^0, 2^6]\}, \tag{5.13}$$

$$\mathrm{gamma} \quad = \{\log_{10}(x) \mid x \in [10^{-3}, 10^0]\}. \tag{5.14}$$

The values assigned to C and gamma are tuned to avoid over-fitting/under-fitting the training set, the best results are selected by means of the *F-score* [54], corresponding to the harmonic mean of *precision* and *recall*.

5.4.2 Double compression identification

Constant Bit Rate Figure 5.15 compares in a CBR scenario the video configurations in Table 5.2 thus describing the double identification performances by means of ROC curves. In Figure 5.15 we compare: the original algorithm of the VPF, therefore without B-frames; the VPF algorithm

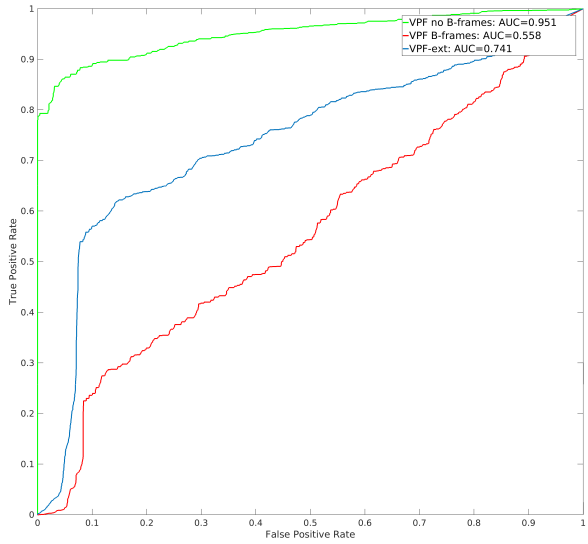


Figure 5.15: ROC double compression identification in a CBR scenario.

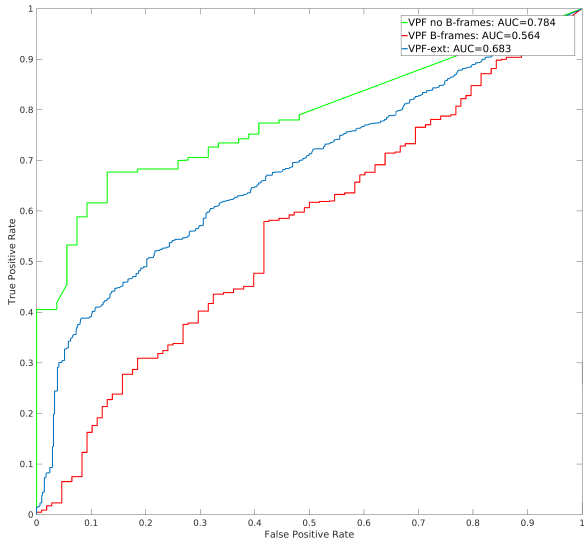


Figure 5.16: ROC double compression identification in a VBR scenario.

considering B-frames and finally the VPF-ext algorithm proposed in this Chapter .

As we extensively described, the effect of introducing B-frames in the second compression drifts the VPF-effect into the next P-frame, causing the loss of periodicity and resulting in an Area Under the Curve of 0.55, whereas the new approach increases the performance of the original VPF of 0.2 in this preliminary tests.

Variable Bit Rate Figure 5.16 depicts the VBR performances of the proposed method by comparing it with the VPF without B-frames and with B-frames. It is interesting noting that these preliminary results show that in case of a variable bit rate, the VPF-ext does not perform well, indeed an AUC of 0.68 is obtained, but we have to consider the fact that cannot perform better that the original VPF, since this extension uses the VPF-period and corrects it when B-frames occur. A clear limitation that has to be further improved. It is interesting to evaluate the same data but to focus on

the number of consecutive B-frames; the results show that an AUC of 0.70 can be obtained if considering $n_B = 3$ but on the other hand with $n_B = 5$ the AUC drops at 0.66, that is probably due to the fact that the number of sub-GOPs in the first case are much more that the second case, thus more samples can improve the overall classification.

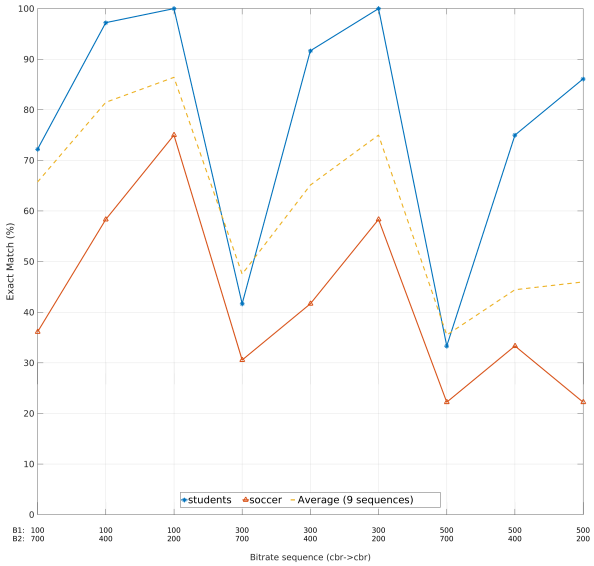
5.4.3 GOP estimation

The GOP estimation of the proposed algorithm has been tested in both Constant and Variable Bit Rate scenarios, in addition a comparison with a state of the art work by *Bestagini et al.* [15] has been considered, just for the VBR case, since the authors of [15] do not argue the CBR case. The reason behind this choice is due to the fact that many state-of-the-art algorithms that study the GOP estimation of a double compression problem do not take into consideration the B-frame scenario, as discussed deeply in Chapter 2.

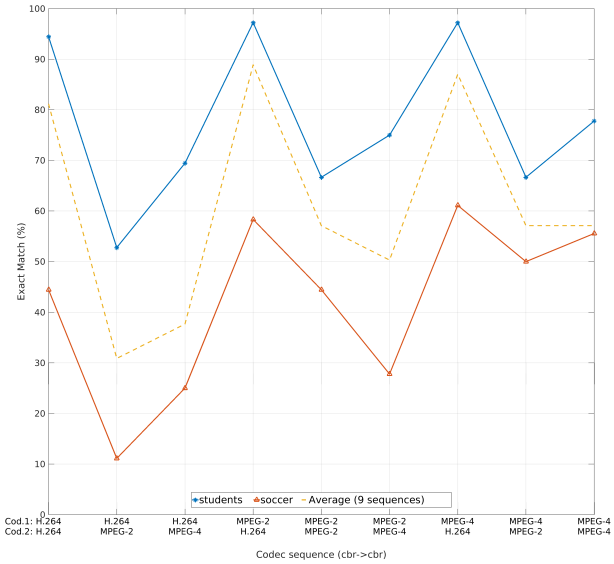
Constant Bit Rate Figure 5.17 depicts the VPF-ext performances with 9 double compressed videos in CBR. Figure 5.17(a) focuses on the Bit-rate values between the first and the second compression, and reports the exact GOP match for each encoding algorithm and GOP size. The best results are obtained when the first compression is less powerful then the second one, thus leaving more traces to build the VPF on. Figure 5.17(b) describes the same data, but it focus on the encoding algorithm pair between the two compression stages. The best results are obtained when H.264/AVC belongs to the second compression, whereas the pair (H.264, MPEG-2) has the worst performances, reasonable due to the less-refined characteristics of the MPEG-2 compression algorithm with respect to the H.264 one.

Variable Bit Rate Figure 5.18 shows the VPF-ext performances with VBR, and Figure 5.19 represents the same data analysed by *Bestagini et al.* [15]. It is important to notice that VPF-ext technique outperforms *Bestagini et al.* on the computational cost, indeed a single video GOP estimation is performed in a few seconds from the proposed algorithm whereas *Bestagini et al.* takes several minutes.

Figure 5.18(a) and Figure 5.19(a) describe the exact GOP match via the quantization parameters used in the first and second encoding procedures. Both algorithms perform similarly when the second compression is lighter, although the VPF-ext obtains better results if all the quantization parameters



(a) Comparison with respect to Bitrate values.



(b) Comparison with respect to the coding algorithm.

Figure 5.17: Double compression GOP estimation in a CBR scenario.

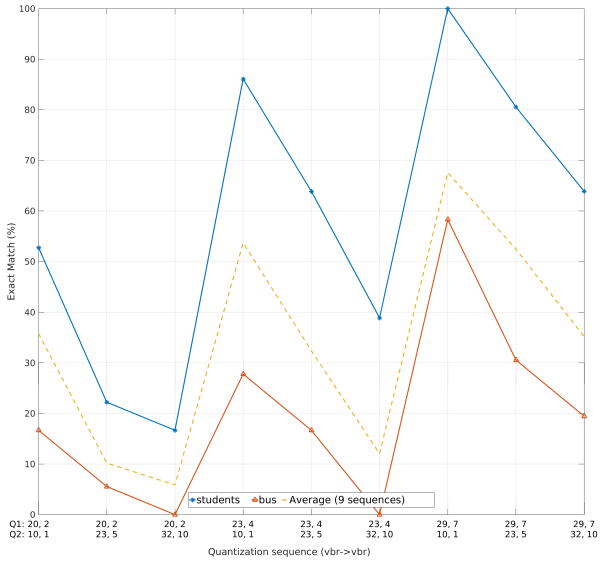
are taken under consideration. It is interesting noticing the performances obtained from the video *Bus*, the worst case scenario for both algorithms, probably due to the video itself that is recorded with a zoom-out and a subsequent pan movement, the rest of the videos in dataset do not share these features.

The VPF-ext method outdoes on average *Bestagini et al.* method, but gets the same poor performances on the $(20, 32)$ and $(23, 32)$ quantization pairs as can be expected since the second compression in quite strong and weakens the traces studied by both methods. In Figure 5.18(b) and Figure 5.19(b) both methods are evaluated by focusing on the coding algorithm. The depicted results confirm the analysis argued so far, the average VPF-ext performances correspond to the best video in *Bestagini et al.* [15].

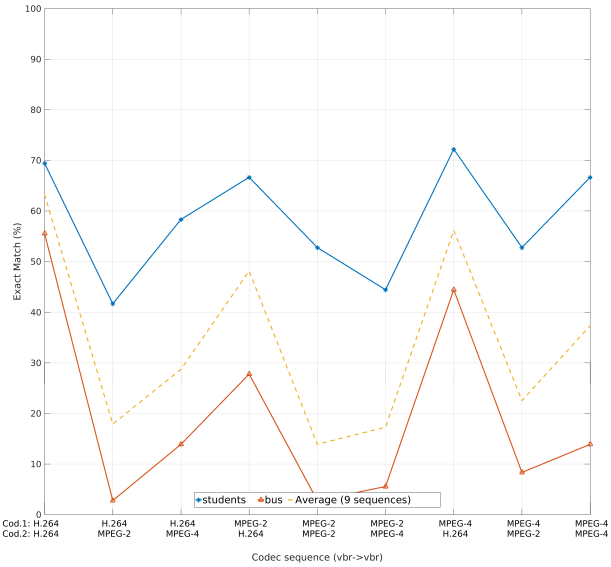
5.5 Remarks

This Chapter proposes an extension of the Variation of Prediction Footprint introduced by *Vázquez-Padín et al.* [65]. The proposed method extends the VPF by introducing to the analysis the presence of B-frames while motivating the VPF-effect by means of Rate-Distortion curves in compressions that include P-frames and B-frames. We discussed that the VPF-effect manifests in the first P-frame after the sub-GOP B, and we proposed an SVM framework to correct this drift, thus identifying the correct first GOP. The proposed approach has been tested in a configuration that uses CBR and VBR, and the performances are reported for the double compression identification with respects to the original VPF, and by considering *Bestagini et al.* as a comparative algorithm to test the GOP estimation in VBR. The overall results show the benefits of this extended approach on the GOP estimation and the overall computational cost.

As a future work, we propose to extend the test to a real acquisition scenario, as the one introduced by the VISION dataset in Chapter 3, and evaluate the possibility of including in the feature matrix the contribution of P-MBs that uses zero Motion Vectors, that to a preliminary analysis seems to behave as S-MBs. Therefore, the **Clf+** kernel can be substituted by an *RBF* one, that will increase the computational cost of the training but can probably identify a better classification hyperplane.

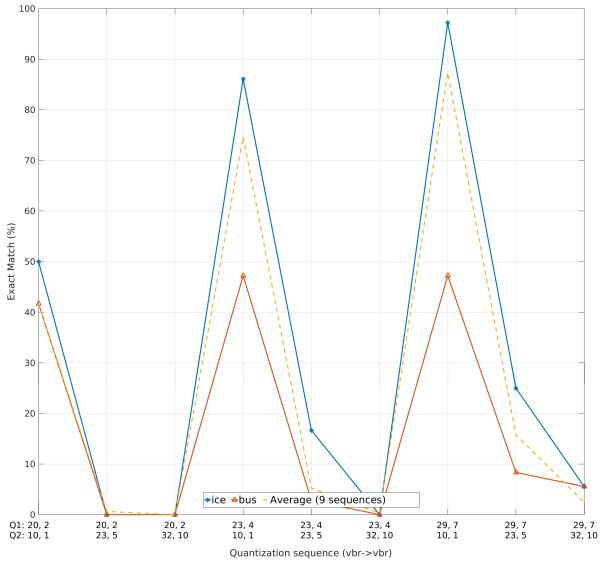


(a) Comparison with respect to Quantization values.

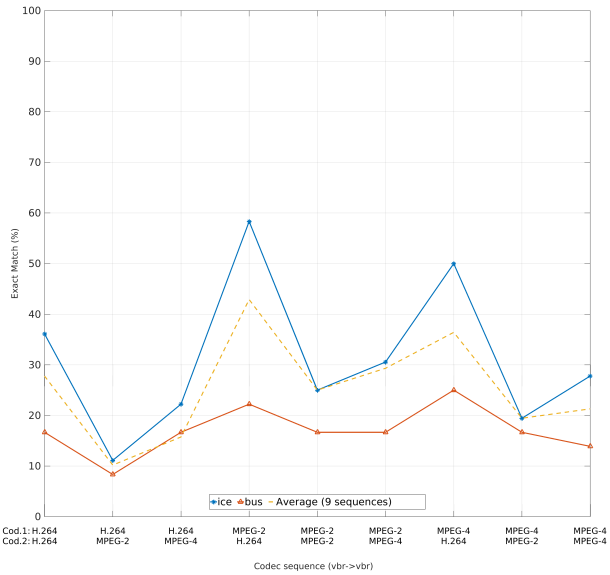


(b) Comparison with respect to the coding algorithm.

Figure 5.18: Double compression GOP estimation in a VBR scenario.



(a) Comparison with respect to Quantization values.



(b) Comparison with respect to the coding algorithm.

Figure 5.19: Double compression GOP estimation in a VBR scenario performed by *Bestagini et al.* [15].

Chapter 6

Conclusion

This Chapter summarizes the contribution of the thesis and discusses avenues for future research.

6.1 Summary of contribution

In this thesis we discussed multimedia forensics issues, such as video origin identification, classification and integrity verification, thus introducing three contributions to solve them with.

- In Chapter 3 a novel Dataset have been introduced. With respect to the state of the art, VISION includes videos and images captured from to 35 modern devices belonging to 11 different brands, and provides for each content the respective *social version*, the one uploaded and downloaded from the main social platforms namely Facebook, Youtube and WhatsApp. Furthermore, some forensics tools have been tested on VISION, thus highlighting interesting issues that have not been noted until now, since there were no sizeable benchmarks available in the research community.
- In Chapter 4 we introduced a new forensic tool that exploits, Low-level video features extracted from the video container structure and content. We argued that these features are able to perform integrity verification on video contents coming from VISION and altered for example by *FFmpeg* or *Exiftool*. Furthermore, these low-level characteristics have been exploited to distinguish the acquisition source, in

terms of Brand-Model, thus achieving source identification and classification.

- In Chapter 5 we proposed a novel extension on the Variation of Prediction Footprint in collaboration with the authors in [65]. The new approach, extended to double compressed scenarios with B-frames, introduced a mathematical formulation based on Rate-Distortion curves that explains the behaviour of VPF effect, and provided a prediction framework based on SVM to correct the signal that locates the GOP of the first compression in cases of Constant Bit Rate and Variable Bit Rate encodings.

6.2 Directions for future work

The introduction of the new dataset can be further expanded into two main directions: the first being the inclusion of new devices by means of the Mobile Application (MOSES [58]), currently available for devices running Android, but still in Beta for iOS mobiles; the second direction would be to consider including altered digital contents as we did in Chapter 4, thus providing a broader dataset for tampering detection.

The VPF-ext algorithm, introduced in Chapter 5 can be improved by considering three aspects:

- extend the number of extracted features
- introduce a coding-based feature extraction
- substitute the classification algorithm

For what concerns the first aspect, the addition of other features, it would be interesting evaluating the VPF effect on B-frames motion vectors then adjusting the feature extraction on a microscopic level, meaning to take into consideration when and how a specific macroblock changes the prediction type instead of having just the frame cardinality for each prediction type.

The current feature extraction, by means of *FFmpeg* dump can be speeded up by developing an algorithm that operates directly within the encoding procedure, thus decrementing the overall time cost and consequently exploiting the VPF-effect on the same macroblock prediction in time and space.

An interesting future activity to the proposed algorithm is to replace or combine the SVM classifier, a simple solution to achieve higher trade

off between classification-power and computational-cost, with other machine learning techniques such as CNN - Convolutional Neural Network that would take as input the prediction features collected in the proposed algorithm.

Appendix A

Appendix

This appendix is related to the method presented in Chapter 4, we discuss in Section A.1 the Likelihood metric and the main issues that can affect it, in Section A.2 we provide an example of a full container structure and container dissimilarities between videos from the same brand.

A.1 Likelihood metric discussion

When calculating the likelihood ratio as in equation (4.7) some special cases may occur; in the following, we explain how we implemented the system to tackle with them. Moreover, we provide a way to account for correlated features that violate the independence assumption requested by equation (4.7).

Null Denominator

When $W_{\overline{C}}(\omega_j) = 0$, the ratio takes the form of $\frac{w_i}{0}$, with w_i the weight associated with the attribute value for the class C. To overcome this issue, we modified the computation to

$$\frac{w_i}{\frac{1}{N_C + 1}}$$

where N_C is the number of videos in the class C. In this way, the denominator will have the smallest possible value and the ratio will still pull the likelihood towards the class C.

Null numerator

When $W_{\overline{C}}(\omega_j) = 0$, the ratio takes the form of $\frac{0}{w_j}$, with w_j the weight associated with the attribute value for the class $\overline{C}$. In this case, since the numerator is zero, the ratio will equal to zero too. This is an unwanted scenario because, since all the ratios are multiplied between them, a single ratio that equals to zero is enough to make the general likelihood zero too. We solve this problem by modifying the computation of the ratio for this case to

$$\frac{\frac{1}{N_{\overline{C}}+1}}{w_j}$$

where $N_{\overline{C}}$ is the number of videos in the class $\overline{C}$.

Null numerator and numerator

When an atom or an attribute value of the query file container is not present in the configuration files, we have the case where both weights equal to zero. Since our goal is to determine whether a given query video X belongs to the class C, any information of its file container that is not present in the configuration file associated with the class C will be treated as if the video does not belong to the class C. That is the case even if the same information is also not present in the configuration file for the class $\overline{C}$. Therefore, we solve this issue by treating it as the null numerator case.

Correlated Features

When multiplying the ratios for a specified atom, the resulting value can be seen as a likelihood of observing that particular atom with its attributes values in either of the two classes. However, the values of the attributes for an atom are not, in general, independently distributed. For example, the values of some attribute can change by groups. If we only multiplied all the ratios, it could be as if we are counting the same information multiple times, pushing the likelihood towards the class C or the class $\overline{C}$ improperly.

To solve this issue, we multiply the likelihood ratios of each atom by also taking into consideration the decorrelation factors. Given a vector of

likelihood ratios $(x_1, \ldots, x_n)$ we compute the likelihood

$$L(\overline{x}) = \prod_{i=1}^{n} x_i^{\alpha_i}$$

with

$$\alpha_i = \frac{(n-1)\gamma_i + 1}{n}$$

$$\gamma_i = -\frac{n}{logn} P(x_i) log P(x_i)$$

and where $P(x_i)$ represents the probability of finding that value of ratio in the vector. Thus, we may have three cases:

1) $P(x_i) = \frac{1}{n}$ with $x_i \neq x_j$, $\forall i \neq j$. Then we have

$$\gamma_i = -\frac{n}{logn} \frac{1}{n} log \frac{1}{n},$$

resulting in $\alpha_i = 1$. The likelihood will be computed as

$$L(\overline{x}) = \prod_{i=1}^{n} x_i.$$

2) $P(x_i) = 1$ with $x_i = x_j$, $\forall i, j$. Then we have $\gamma_i = 0$ and $\alpha_i = \frac{1}{n}$. The likelihood will be computed as

$$L(\overline{x}) = \prod_{i=1}^{n} x_i^{\frac{1}{n}} = x_1^{\sum \frac{1}{n}} = x_1.$$

3) $x_i = x_j$, $i, j = 1, \ldots, k$ with $P(x_i) = \frac{k}{n}$ for $i = 1, \ldots, k$ and $P(x_j) = \frac{1}{n}$ for $j > k$. Then we have

$$\gamma_i = -n \frac{n}{logn} \frac{k}{n} log \frac{k}{n} = k(log \frac{k}{n}) logn,$$

and

$$\alpha_i = \frac{(n-1)(klog \frac{k}{n}) logn + 1}{n}.$$

The likelihood will be computed as

$$L(\overline{x}) = x_i^{\sum \frac{1}{n}} = x_1^{\left[(n-1)(klog \frac{k}{n}) logn + 1 \right] \frac{k}{n}} \cdot \prod_{j=k+1}^{n} x_j.$$

A.2 MP4-like container discussion

Full container structure

In Listing A.1 it is reported the full container structure belonging to a native video content acquired with a *Samsung Galaxy S3* as an extension to the fragment reported in Figure 4.1.

Listing A.1: *Samsung Galaxy S3* sample video container

```
<?xml version="1.0" encoding="UTF-8"?>
<root modelName="phoneBrandName">
  <ftyp-1 majorBrand="isom" minorVersion="0" compatibleBrand_1="
      isom" compatibleBrand_2="3gp4" count="0" />
  <moov-2 stuff="MovieBox[]" count="0">
    <mvhd-1 creationTime="Fri Aug 07 13:00:15 CEST 2015"
        modificationTime="Fri Aug 07 13:00:15 CEST 2015"
        timescale="1000" duration="72856" rate="1.0" volume="1.0
        " matrix="Rotate 0" nextTrackId="3" version="0" flags="0
        " count="0" />
    <trak-2 stuff="TrackBox[]" count="0">
      <tkhd-1 creationTime="Fri Aug 07 13:00:15 CEST 2015"
          modificationTime="Fri Aug 07 13:00:15 CEST 2015"
          trackId="1" duration="72619" layer="0" alternateGroup=
          "0" volume="0.0" matrix="Rotate 0" width="1920.0"
          height="1080.0" version="0" flags="7" count="0" />
      <mdia-2 stuff="MediaBox[]" count="0">
        <mdhd-1 creationTime="Fri Aug 07 13:00:15 CEST 2015"
            modificationTime="Fri Aug 07 13:00:15 CEST 2015"
            timescale="90000" duration="6535724" language="```"
            version="0" flags="0" count="0" />
        <hdlr-2 handlerType="vide" name="VideoHandle" version="0
            " flags="0" count="0" />
        <minf-3 stuff="MediaInformationBox[]" count="0">
          <vmhd-1 graphicsmode="0" opcolor0="0" opcolor1="0"
              opcolor2="0" version="0" flags="1" count="0" />
          <dinf-2 stuff="DataInformationBox[]" count="0">
            <dref-1 version="0" flags="0" count="0">
              <url-1 stuff="DataEntryUrlBox[]" count="0" />
            </dref-1>
          </dinf-2>
          <stbl-3 stuff="SampleTableBox[]" count="0">
            <stsd-1 version="0" flags="0" count="0">
              <avc1-1 data_reference_index="1" width="1920"
                  height="1080" horizresolution="72.0"
                  vertresolution="72.0" frame_count="1"
                  compressorname="null" depth="24" count="0">
                <avcC-1 configurationVersion="1"
                    avcProfileIndication="66"
                    profileCompatibility="128"
                    avcLevelIndication="40" lengthSizeMinusOne="
                    3" hasExts="false" chromaFormat="-1"
                    bitDepthLumaMinus8="-1" bitDepthChromaMinus8
                    ="-1" lengthSizeMinusOnePaddingBits="63"
```

```
                    numberOfSequenceParameterSetsPaddingBits="7"
                     chromaFormatPaddingBits="31"
                    bitDepthLumaMinus8PaddingBits="31"
                    bitDepthChromaMinus8PaddingBits="31" count="
                    0" />
                <pasp-2 hSpacing="65536" vSpacing="65536" count=
                    "0" />
              </avc1-1>
            </stsd-1>
            <stts-2 entryCount="1468" version="0" flags="0"
                count="0" />
            <stss-3 entryCount="73" version="0" flags="0" count=
                "0" />
            <stsz-4 sampleSize="0" sampleCount="2179" version="0
                " flags="0" count="0" />
            <stsc-5 entryCount="3" version="0" flags="0" count="
                0" />
            <co64-6 entryCount="71" version="0" flags="0" count=
                "0" />
          </stbl-3>
        </minf-3>
      </mdia-2>
    </trak-2>
    <trak-3 stuff="TrackBox[]" count="0">
      <tkhd-1 creationTime="Fri Aug 07 13:00:15 CEST 2015"
          modificationTime="Fri Aug 07 13:00:15 CEST 2015"
          trackId="2" duration="72856" layer="0" alternateGroup=
          "0" volume="1.0" matrix="Rotate 0" width="0.0" height=
          "0.0" version="0" flags="7" count="0" />
      <mdia-2 stuff="MediaBox[]" count="0">
        <mdhd-1 creationTime="Fri Aug 07 13:00:15 CEST 2015"
            modificationTime="Fri Aug 07 13:00:15 CEST 2015"
            timescale="48000" duration="3497069" language="```"
            version="0" flags="0" count="0" />
        <hdlr-2 handlerType="soun" name="SoundHandle" version="0
            " flags="0" count="0" />
        <minf-3 stuff="MediaInformationBox[]" count="0">
          <smhd-1 balance="0.0" version="0" flags="0" count="0"
              />
          <dinf-2 stuff="DataInformationBox[]" count="0">
            <dref-1 version="0" flags="0" count="0">
              <url-1 stuff="DataEntryUrlBox[]" count="0" />
            </dref-1>
          </dinf-2>
          <stbl-3 stuff="SampleTableBox[]" count="0">
            <stsd-1 version="0" flags="0" count="0">
              <mp4a-1 bytesPerSample="0" bytesPerFrame="0"
                  bytesPerPacket="0" samplesPerPacket="0"
                  packetSize="0" compressionId="0" soundVersion=
                  "0" sampleRate="48000" sampleSize="16"
                  channelCount="2" count="0">
                <esds-1 esId="0" streamDependenceFlag="0"
                    URLFlag="0" oCRstreamFlag="0" streamPriority
                    ="0" URLLength="0" URLString="'null'"
                    remoteODFlag="0" dependsOnEsId="0" oCREsId="
                    0" objectTypeIndication="64" streamType="5"
```

```
                upStream="0" bufferSizeDB="768" maxBitRate="
                196000" avgBitRate="196000"
                decoderSpecificInfo="null" configBytes="1190
                " audioObjectType="2 (AAC LC)"
                samplingFrequencyIndex="3 (48000)"
                samplingFrequency="0" channelConfiguration="
                2" syncExtensionType="-1" frameLengthFlag="0
                " dependsOnCoreCoder="0" coreCoderDelay="0"
                extensionFlag="0" layerNr="0" numOfSubFrame=
                "0" layer_length="0"
                aacSectionDataResilienceFlag="false"
                aacScalefactorDataResilienceFlag="false"
                aacSpectralDataResilienceFlag="false"
                extensionFlag3="0" configDescriptorDeadBytes
                ="null" profileLevelIndicationDescriptors="
                null" predefined="2" count="0" />
            </mp4a-1>
          </stsd-1>
          <stts-2 entryCount="248" version="0" flags="0" count
              ="0" />
          <stsz-3 sampleSize="0" sampleCount="3415" version="0
              " flags="0" count="0" />
          <stsc-4 entryCount="3" version="0" flags="0" count="
              0" />
          <co64-5 entryCount="73" version="0" flags="0" count=
              "0" />
        </stbl-3>
      </minf-3>
    </mdia-2>
  </trak-3>
</moov-2>
<free-3 stuff="com.coremedia.iso.boxes.FreeBox@1" count="0" />
<mdat-4 size="20593115" count="0" />
</root>
```

Brand comparison

In Listing A.2 is reported the container comparison between two videos an *Apple iPhone 4s* and an *Apple iPhone 5c* showing the similarities and dissimilarities that the integrity verification method described in 4.3 is able to produce. It is clear that the two containers are very similar, indeed 90% of the container is shared whereas just a 10% is discriminatory, with this feature we are confident that the two videos belong to the same brand. On the other hand, if videos belonging to different brands are compared, usually the percentage of similarities is about 1% of the container whereas 99% of it is different, for simplicity we omit an example of such scenario.

Listing A.2: Container comparison between videos from the same brand.

```
{"reference":"D02_Apple_iPhone4s/D02_V_flat_move_0001.mov",
```

```
"query":"D05_Apple_iPhone5c/D05_V_flat_move_0001.mov",
"tot":285,"diff":30,"fields":
[{"field":"nextTrackId","queryValue":"5",
"atom":"mvhd-1","refValue":"3"},
{"field":"volume","queryValue":"1.0","atom":"tkhd-1",
"refValue":"0.0"},
{"field":"matrix","queryValue":"Rotate 0","atom":"tkhd-1",
"refValue":"u:0.0 v:0.0 w:1.0 a:-1.0 b:0.0 c:0.0 d:-1.0 tx:1920
    .0 ty:1080.0"},
{"field":"segmentDuration","queryValue":"42758","atom":"elst-1",
    "refValue":"35800"},
{"field":"avcProfileIndication","queryValue":"100",
"atom":"avcC-1","refValue":"66"},
{"field":"hasExts","queryValue":"true","atom":"avcC-1",
"refValue":"false"},
{"field":"chromaFormat","queryValue":"1","atom":"avcC-1",
"refValue":"-1"},
{"field":"bitDepthLumaMinus8","queryValue":"0",
"atom":"avcC-1","refValue":"-1"},
{"field":"bitDepthChromaMinus8","queryValue":"0",
"atom":"avcC-1","refValue":"-1"},
{"field":"chromaFormatPaddingBits","queryValue":"63",
"atom":"avcC-1","refValue":"31"},
{"field":"version","queryValue":"null",
"atom":"meta-5","refValue":"0"},
{"field":"flags","queryValue":"null",
"atom":"meta-5","refValue":"0"},
{"field":"count","queryValue":"null",
"atom":"meta-5","refValue":"0"},
{"field":"handlerType","queryValue":"null",
"atom":"hdlr-1","refValue":"mdta"},
{"field":"name","queryValue":"null","atom":"hdlr-1",
"refValue":"null"},
{"field":"version","queryValue":"null",
"atom":"hdlr-1", "refValue":"0"},
{"field":"flags","queryValue":"null","atom":"hdlr-1",
"refValue":"0"},
{"field":"count","queryValue":"null","atom":"hdlr-1",
"refValue":"0"},
{"field":"count","queryValue":"null",
"atom":"keys-2","refValue":"0"},
{"field":"count","queryValue":"null","atom":"ilst-3",
"refValue":"0"},
{"field":"count","queryValue":"null",
"atom":"unkn-1","refValue":"0"},
{"field":"count","queryValue":"null","atom":"unkn-2",
"refValue":"0"},
{"field":"segmentDuration","queryValue":"42758",
"atom":"elst-1","refValue":"35799"},
{"field":"count","queryValue":"null",
"atom":"udta-4","refValue":"0"},
{"field":"count","queryValue":"null","atom":"mod-1",
"refValue":"0"},
{"field":"count","queryValue":"null",
"atom":"swr-2","refValue":"0"},
{"field":"count","queryValue":"null","atom":"day-3",
```

```
    "refValue":"0"},
    {"field":"count","queryValue":"null",
    "atom":"mak-5","refValue":"0"},
    {"field":"count","queryValue":"null","atom":"free-5",
    "refValue":"0"},
    {"field":"count","queryValue":"null",
    "atom":"free-4","refValue":"0"}]}
```

Appendix B

Publications

This research activity has led to several publications in international journals and conferences. These are summarized below.[1]

International Journals

1. Bianchi, T., Piva, A., and **Shullani, D.**. "Anticollusion solutions for asymmetric fingerprinting protocols based on client side embedding." *EURASIP Journal on Information Security*, 2015. *4 citations*

1. **Shullani, D.**, Fontani, M., Iuliani, M., Al Shaya, O. and Piva, A. "VISION: a video and image dataset for source identification", *EURASIP Journal on Information Security*, 2017. *1 citation*

National Conferences

1. **Shullani, D.**, Al Shaya, O., Iuliani, M., Fontani, M. and Piva, A. " A Dataset for Forensic Analysis of Videos in the Wild." In International Tyrrhenian Workshop on Digital Communication (pp. 84-94), *Springer, Cham*, Italy, September 2017.

Technical Reports

1. Iuliani, M., Fontani, M., **Shullani, D.** and Piva, A. "A Hybrid Approach to Video Source Identification", *arXiv preprint* arXiv:1705.01854, 2017.

[1]The author's bibliometric indices are the following: H-index = 1, total number of citations = 5 (source: Google Scholar on Month 10, 2017).

To be Submitted

1. "**Forensic Analysis of Video File Containers**", Iuliani, M., **Shullani D.**, Fontani, M., Meucci, S. and Piva, A., "to be submitted to", *IEEE Transactions on Information Forensics and Security*, 2018.

1. "**Forgery detection with a Variation of Prediction Footprint extension**", Vazquez-Padin, D., Fontani, M., **Shullani, D.**, Piva, A., Perez-Gonzales, F., and Barni, M. "to be submitted to", *IEEE Transactions on Information Forensics and Security*, 2018.

Bibliography

[1] "Clip grab," 2017. [Online]. Available: https://clipgrab.org/

[2] I. 10918-1, "Information technology. digital compression and coding of continuous-tone still images," *ITU-T Recommendation T.81*, 1992.

[3] I. 14496, "Information technology. coding of audio-visual objects, part 14: Mp4 file format," 2003.

[4] ——, "Information technology. coding of audio-visual objects, part 12: Iso base media file format, 3rd ed." 2008.

[5] J. Abbasi Aghamaleki and A. Behrad, "Malicious inter-frame video tampering detection in mpeg videos using time and spatial domain analysis of quantization effects," *Multimedia Tools and Applications*, vol. 76, no. 20, pp. 20 691–20 717, Oct 2017. [Online]. Available: https://doi.org/10.1007/s11042-016-4004-z

[6] J. A. Aghamaleki and A. Behrad, "Inter-frame video forgery detection and localization using intrinsic effects of double compression on quantization errors of video coding," *Signal Processing: Image Communication*, vol. 47, pp. 289–302, 2016.

[7] C. Aitken, A. Barrett, C. Berger, A. Biedermann, C. Champod, T. Hicks, J. Lucena-Molina, L. Lunt, S. McDermott, L. McKenna *et al.*, "Enfsi guideline for evaluative reporting in forensic science," *European Network of Forensic Science Institutes (ENFSI)*, 2015.

[8] O. I. Al-Sanjary, A. A. Ahmed, and G. Sulong, "Development of a video tampering dataset for forensic investigation," *Forensic Science International*, vol. 266, pp. 565–572, 2016.

[9] Apache, "Java mp4 parser," 2017. [Online]. Available: http://www.github.com/sannies/mp4parser

[10] I. Apple Computer, "Quicktime file format," 2018. [Online]. Available: https://developer.apple.com/library/content/documentation/QuickTime/QTFF/QTFFPreface/qtffPreface.html#//apple_ref/doc/uid/TP40000939

[11] F. Benford, "The law of anomalous numbers," *Proceedings of the American Philosophical Society*, vol. 78, no. 4, pp. pp. 551–572, 1938.

[12] F. Bertini, R. Sharma, A. Iannı, D. Montesi, and M. A. Zamboni, "Social media investigations using shared photos," in *The International Conference on Computing Technology, Information Security and Risk Management (CTISRM2016)*, 2016, p. 47.

[13] P. Bestagini, A. Allam, S. Milani, M. Tagliasacchi, and S. Tubaro, "Video codec identification," in *Proc. of 2012 International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, Mar. 25 – 30, 2012, pp. 2257–2260.

[14] P. Bestagini, S. Milani, M. Tagliasacchi, and S. Tubaro, "Local tampering detection in video sequences," in *2013 IEEE 15th International Workshop on Multimedia Signal Processing (MMSP)*, Sept 2013, pp. 488–493.

[15] ——, "Codec and gop identification in double compressed videos," *IEEE Transactions on Image Processing*, vol. 25, no. 5, pp. 2298–2310, 2016.

[16] D. Bolton, "youtube-dl documentation," 2017. [Online]. Available: github.com/rg3/youtube-dl/blob/master/README.md#readme

[17] B. E. Boser, I. M. Guyon, and V. N. Vapnik, "A training algorithm for optimal margin classifiers," in *Proceedings of the fifth annual workshop on Computational learning theory*. ACM, 1992, pp. 144–152.

[18] M. Chen, J. Fridrich, M. Goljan, and J. Lukáš, "Source digital camcorder identification using sensor photo response non-uniformity," in *Electronic Imaging 2007*. International Society for Optics and Photonics, 2007, pp. 65 051G–65 051G.

[19] ——, "Determining image origin and integrity using sensor noise," *Information Forensics and Security, IEEE Transactions on*, vol. 3, no. 1, pp. 74–90, 2008.

[20] S. Chen, T. Sun, X. Jiang, P. He, S. Wang, and Y. Q. Shi, "Detecting double h. 264 compression based on analyzing prediction residual distribution," in *International Workshop on Digital Watermarking*. Springer, 2016, pp. 61–74.

[21] S. Chen, A. Pande, K. Zeng, and P. Mohapatra, "Live video forensics: Source identification in lossy wireless networks," *IEEE Transactions on Information Forensics and Security*, vol. 10, no. 1, pp. 28–39, 2015.

[22] W.-H. Chuang, H. Su, and M. Wu, "Exploring compression effects for improved source camera identification using strongly compressed video," in *IEEE International Conference on Image Processing (ICIP)*. IEEE, 2011, pp. 1953–1956.

[23] C. Cortes and V. Vapnik, "Support-vector networks," *Machine Learning*, vol. 20, no. 3, pp. 273–297, 1995.

[24] A. Costanzo and M. Barni, "Detection of double avc/hevc encoding," in *Signal Processing Conference (EUSIPCO), 2016 24th European.* IEEE, 2016, pp. 2245–2249.

[25] D.-T. Dang-Nguyen, C. Pasquini, V. Conotter, and G. Boato, "Raise: A raw images dataset for digital image forensics," in *Proceedings of the 6th ACM Multimedia Systems Conference*, ser. MMSys '15. New York, NY, USA: ACM, 2015, pp. 219–224.

[26] A. De Rosa, A. Piva, M. Fontani, and M. Iuliani, "Investigating multimedia contents," in *2014 International Carnahan Conference on Security Technology (ICCST)*, Oct 2014, pp. 1–6.

[27] J. Electronics and I. T. I. A. J. CP-3451, "Exchangeable image file format for digital still cameras: Exif version 2.2," 2002.

[28] H. Farid, "Digital image ballistics from jpeg quantization: a followup study," *[Technical Report TR2008–638] Department of Computer Science, Dartmouth College, Hanover, NH, USA*, 2008.

[29] A. Gironi, M. Fontani, T. Bianchi, A. Piva, and M. Barni, "A video forensic technique for detecting frame deletion and insertion," in *Acoustics, Speech and Signal Processing (ICASSP), 2014 IEEE International Conference on.* IEEE, 2014, pp. 6226–6230.

[30] T. Gloe, "Forensic analysis of ordered data structures on the example of jpeg files," in *Information Forensics and Security (WIFS), 2012 IEEE International Workshop on.* IEEE, 2012, pp. 139–144.

[31] T. Gloe and R. Böhme, "The Dresden image database for benchmarking digital image forensics," *Journal of Digital Forensic Practice*, vol. 3, no. 2-4, pp. 150–159, 2010.

[32] T. Gloe and R. Böhme, "The 'Dresden Image Database' for benchmarking digital image forensics," in *Proceedings of the 25th Symposium On Applied Computing (ACM SAC 2010)*, vol. 2, 2010, pp. 1585–1591.

[33] T. Gloe, A. Fischer, and M. Kirchner, "Forensic analysis of video file formats," *Digital Investigation*, vol. 11, Supplement 1, pp. S68 – S76, 2014, proceedings of the First Annual {DFRWS} Europe. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1742287614000140

[34] M. Goljan and J. Fridrich, "Camera identification from scaled and cropped images," *Security, Forensics, Steganography, and Watermarking of Multimedia Contents X*, vol. 6819, p. 68190E, 2008.

[35] M. Goljan, J. Fridrich, and T. Filler, "Large scale test of sensor fingerprint camera identification," in *IS&T/SPIE Electronic Imaging*. International Society for Optics and Photonics, 2009, pp. 72 540I–72 540I.

[36] T. E. Group, "The inclusive internet: Mapping progress 2017," 2017. [Online]. Available: https://theinclusiveinternet.eiu.com/

[37] P. He, X. Jiang, T. Sun, and S. Wang, "Double compression detection based on local motion vector field analysis in static-background videos," *Journal of Visual Communication and Image Representation*, vol. 35, pp. 55–66, 2016.

[38] M. Huang, R. Wang, J. Xu, D. Xu, and Q. Li, *Detection of Double Compression for HEVC Videos Based on the Co-occurrence Matrix of DCT Coefficients*. Cham: Springer International Publishing, 2016, pp. 61–71. [Online]. Available: https://doi.org/10.1007/978-3-319-31960-5_6

[39] S. Inc., "Statista," 2017. [Online]. Available: http://www.statista.com/statistics/263437/global-smartphone-sales-to-end-users-since-2007/

[40] ISO/IEC, "Information technology - Coding of audio-visual objects - Part 2: Visual," International Organization for Standardization, Geneva, Switzerland, ISO/IEC 14496-2, 2004.

[41] ——, "Information technology - generic coding of moving pictures and associated audio information - Part 2: Video," International Organization for Standardization, Geneva, Switzerland, ISO/IEC IS 13818-2, 2008.

[42] ITU-T/ISO/IEC, "Advanced Video Coding for Generic Audio-Visual Services," International Telecommunication Union and International Organization for Standardization, Geneva, Switzerland, Tech. Rep. ITU-T Rec. H.264 and ISO/IEC 14496-10 (AVC), 2003.

[43] ——, "High Efficiency Video Coding (HEVC) Text Specification Draft 9," International Telecommunication Union and International Organization for Standardization, ITU-T/ISO/IEC Joint Collaborative Team on Video Coding (JCT-VC), Geneva, Switzerland, ISO/IEC JCTVC-K1003, 2012.

[44] M. Iuliani, M. Fontani, D. Shullani, and A. Piva, "A hybrid approach to video source identification," *arXiv preprint arXiv:1705.01854*, 2017. [Online]. Available: https://arxiv.org/abs/1705.01854

[45] X. Jiang, P. He, T. Sun, F. Xie, and S. Wang, "Detection of double compression with the same coding parameters based on quality degradation mechanism analysis," *IEEE Transactions on Information Forensics and Security*, vol. 13, no. 1, pp. 170–185, 2018.

[46] E. Kee, M. K. Johnson, and H. Farid, "Digital image authentication from jpeg headers," *IEEE Transactions on Information Forensics and Security*, vol. 6, no. 3, pp. 1066–1075, Sept 2011.

[47] S. Knerr, L. Personnaz, and G. Dreyfus, "Single-layer learning revisited: a stepwise procedure for building and training a neural network," *Neurocomputing: algorithms, architectures and applications*, vol. 68, no. 41-50, p. 71, 1990.

[48] P. Korus, "Digital image integrity–a survey of protection and verification techniques," *Digital Signal Processing*, vol. 71, pp. 1–26, 2017.

[49] J. Lukas, J. Fridrich, and M. Goljan, "Digital camera identification from sensor pattern noise," *IEEE Transactions on Information Forensics and Security*, vol. 1, no. 2, pp. 205–214, 2006.

[50] D. Meuwly, D. Ramos, and R. Haraksim, "A guideline for the validation of likelihood ratio methods used for forensic evidence evaluation," *Forensic Science International*, vol. 276, pp. 142 – 153, 2017. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0379073816301359

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

[52] M. Moltisanti, A. Paratore, S. Battiato, and L. Saravo, "Image manipulation on facebook for forensics evidence," in *Image Analysis and Processing - ICIAP 2015 - 18th International Conference, Genoa, Italy, September 7-11, 2015, Proceedings, Part II*, 2015, pp. 506–517.

[53] A. Piva, "An overview on image forensics," *ISRN Signal Processing*, vol. 2013, pp. Article ID 496 701, 22 pages, 2013.

[54] D. M. Powers, "Evaluation: from precision, recall and f-measure to roc, informedness, markedness and correlation," 2011.

[55] G. Qadir, S. Yahaya, and A. T. Ho, "Surrey university library for forensic analysis (sulfa) of video content," 2012.

[56] G. Schaefer and M. Stich, "Ucid: An uncompressed color image database," in *Electronic Imaging 2004*. International Society for Optics and Photonics, 2003, pp. 472–480.

[57] T. Shanableh, "Detection of frame deletion for digital video forensics," *Digital Investigation*, vol. 10, no. 4, pp. 350 – 360, 2013. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1742287613001102

[58] D. Shullani, O. Al Shaya, M. Iuliani, M. Fontani, and A. Piva, "A dataset for forensic analysis of videos in the wild," 2017, to appear In Proceedings of Communications in Computer and Information Science (CCIS).

[59] D. Shullani, M. Fontani, M. Iuliani, O. Al Shaya, and A. Piva, "Vision: a video and image dataset for source identification," *EURASIP Journal on Information Security*, vol. 2017, no. 1, p. 15, 2017.

[60] G. J. Sullivan and T. Wiegand, "Rate-distortion optimization for video compression," *IEEE signal processing magazine*, vol. 15, no. 6, pp. 74–90, 1998.

[61] SWGDE-SWGIT, "Best practices for image content authentication, Version: 1.0," July 2017. [Online]. Available: https://www.swgde.org/documents/CurrentDocuments

[62] SWGIT, "Best practices for image authentication," in *Scientific Working Group on Imaging Technology, Section 14, Version 1.1 2013.01.11.*

[63] S. Taspinar, M. Mohanty, and N. Memon, "Source camera attribution using stabilized video," in *2016 IEEE International Workshop on Information Forensics and Security (WIFS)*, Dec 2016, pp. 1–6.

[64] W. Van Houten and Z. Geradts, "Source video camera identification for multiply compressed videos originating from youtube," *Digital Investigation*, vol. 6, no. 1, pp. 48–60, 2009.

[65] D. Vázquez-Padín, M. Fontani, T. Bianchi, P. Comesaña, A. Piva, and M. Barni, "Detection of video double encoding with GOP size estimation," in *Proc. of 2012 IEEE International Workshop on Information Forensics and Security (WIFS)*, 2012, pp. 151–156.

[66] D. Vazquez-Padin and P. Comesana, "Ml estimation of the resampling factor," in *Information Forensics and Security (WIFS), 2012 IEEE International Workshop on*. IEEE, 2012, pp. 205–210.

[67] D. Vázquez-Padín and F. Pérez-González, "Prefilter design for forensic resampling estimation," in *Proc. of 2011 IEEE International Workshop on Information Forensics and Security (WIFS)*, 2011.

[68] W. Wang and H. Farid, "Exposing digital forgeries in video by detecting double quantization," in *Proc. of the 11th ACM workshop on Multimedia and security (MM&Sec)*, ser. MM&Sec '09. New York, NY, USA: ACM, 2009, pp. 39–48.

[69] 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.

[70] Q. Xu, T. Sun, X. Jiang, and Y. Dong, *HEVC Double Compression Detection Based on SN-PUPM Feature*. Cham: Springer International Publishing, 2017, pp. 3–17. [Online]. Available: https://doi.org/10.1007/978-3-319-64185-0_1

[71] J. Zheng, T. Sun, X. Jiang, and P. He, *Double H.264 Compression Detection Scheme Based on Prediction Residual of Background Regions*. Cham: Springer International Publishing, 2017, pp. 471–482. [Online]. Available: https://doi.org/10.1007/978-3-319-63309-1_43