Computers and Electrical Engineering 81 (2020) 106541

Contents lists available at ScienceDirect

Computers and Electrical Engineering

journal homepage: www.elsevier.com/locate/compeleceng

Efficient embedding and retrieval of information for high-resolution videos coded with HEVC[☆]

D. Rodríguez Galiano[a], A.A. Del Barrio[a,*], G. Botella[a], D. Cuesta[b]

[a] Department of Computer Architecture and Automation, Universidad Complutense de Madrid, Madrid, Spain
[b] Technical Direction, Administrador de Infraestructuras Ferroviarias, Madrid, Spain

ARTICLE INFO

Article history:
Received 14 December 2018
Revised 10 December 2019
Accepted 18 December 2019
Available online 27 December 2019

Keywords:
Steganography
Steganalysis
HEVC
Video coding
Real-time

ABSTRACT

Steganography is the art of hiding information within a file. This work focuses on embedding such information in videos. In this scenario, it is critical to comply with the latest video standard, namely: the High Efficiency Video Coding (HEVC), which allows reducing the size of the file to be transmitted. In this paper we propose an HEVC-compliant method to hide and retrieve information in high-resolution videos. The procedure is based on modifying the luminance of certain blocks, not the HEVC encoder. Nevertheless, this must be carefully done, as the HEVC standard is a powerful attack in itself, since it compresses 50% the size of the video on average and the embedded information may disappear. Results show that it is possible to retrieve all the information while maintaining the quality of the video when using intra frames. Furthermore, the proposed flow has shown to be resilient to state-of-the-art steganalysis attacks.

1. Introduction

Nowadays, video and multimedia content are dominating the Internet. According to a report by Cisco [1], by 2021, global IP traffic will reach 3.3 ZB and online video will be responsible for 82% of this traffic. Furthermore, we are moving towards tremendous resolutions as 4k and 8k, so efficiently employing the bandwidth is a must.

High-Efficiency Video Coding (HEVC) is the latest video coding standard of the Joint Collaborative Team on Video Coding (JCT-VC). HEVC noticeably improves compression performance when compared with previous standards such as H.264 and represents a major step forward in video compression technology. However, this improvement is achieved by increasing the complexity of the encoding process. HEVC employs a novel flexible quad-tree coding block partitioning structure [2] that enables the use of large and multi-sized coding, prediction, and transform blocks. This system is more efficient but also more computationally demanding. The HEVC standard is designed to achieve multiple goals: coding efficiency (in the range of 50%-bit rate reduction for equal perceptual video quality compared with the antecessor H.264/AVC standard), transport system integration and data loss resilience, as well as real-time implementations using parallel processing architectures [2]. Prior to deploying hardware solutions, developers test their approaches to comply with the standard using the reference software HM-16.2 (HEVC Test Model) [3]. This is a simulation software that is being developed by the Joint Collaborative

☆ This paper is for regular issues of CAEE. Reviews processed and recommended for publication to the Editor-in-Chief by Area Editor Dr. M. Malek.
* Corresponding author.
E-mail address: abarriog@ucm.es (A.A. Del Barrio).

https://doi.org/10.1016/j.compeleceng.2019.106541
0045-7906/© 2019 Elsevier Ltd. All rights reserved.

Team in Video Coding (JCT-VC), regrouping experts of ITU-T SG 16 and ISO/IEC SC29 WG11, and provides a basis on which to perform coding experiments in H.265. After testing with HM-16.2, the following step is typically considering x265 [4], which is an open source software able to encode in real-time, as it employs multiple threads.

Besides optimizing the bandwidth use, another great challenge when transmitting information nowadays is ensuring the individuals privacy, as the new General Data Protection Regulation (GDPR) in Europe indicates. Most applications generally rely on cryptographic algorithms like AES to hide the data. Nevertheless, this is only applicable to protect the data while transmitting, but when the content arrives at the destination and is deciphered, it is no longer secure [5]. Thereby, data hiding techniques are able to provide shielding even when data is not ciphered. Such is the case for many multimedia contents that requires to store some data related to authorship or intellectual property.

Steganography is the science branch that tackles the concealing of these data. It appeared many centuries ago, but steganography is still applied as evidence of copyright, to hide sensitive information in medical tests, to anonymize the patients' data or even to improve the analysis of the railroad catenary. In order to efficiently hide information within a video, there are three key features that define the quality of a steganographic system: the attacks resilience, the capacity to embed extra data and the amount of such data that can be recovered.

Nonetheless, the combination of the HEVC standard with steganography is not an easy one. First of all, any coding standard is an attack by itself and, second, the HEVC achieves a high degree of compression, so there is a vast amount of information that is removed with respect to the raw video.

Hence, we propose a method for hiding and recovering data in high-resolution videos compressed with the HEVC standard. The main advantage with respect to prior approaches is that the standard itself is not modified for performing such tasks. Results show that the messages are totally recovered while maintaining the quality of FullHD and 4k videos, for both HM-16.2 and x265 encoding software.

The rest of the paper is organized as follows: Section 2 describes some prior approaches utilizing steganographic techniques in combination with coding standards; Section 3 describes some basic concepts related with video coding and the HEVC standard itself; Sections 4 and 5 explain our proposed methods to embed and retrieve the information, respectively; Sections 6 and 7 present the metrics and the results that verify the applicability and the robustness of the algorithm with steganalysis tools and, lastly, Section 8 gives our final remarks and outlines our future lines of work.

2. Related work

Classical steganographic approaches work directly on the input frame, i.e. the spatial domain. The most widespread example is maybe the LSB algorithm, which directly substitutes the least significant bit of the color components belonging to certain pixels. This approach is undetectable for the human eye, but easily detectable for automated steganalysis techniques.

More robust techniques are based on integrating the data payload in the form of modified coefficients in the frequency domain. For instance, Nakajima et al. [6] propose a method to embed data bits by modifying the coefficients located at certain positions of some 16×16 blocks when employing the MPEG-1 codec. In another work, Wong et al. [7] present a reversible technique for hiding information by modifying the quantized Discrete Cosine Transform (DCT) coefficients at the same time that the quality of the decoded video is maintained. Similar ideas are employed in [8]. Authors in [9-10] describe a watermarking method based on introducing the information within the motions vectors in order to increase the robustness. Xu et al. [11] hide the information by modulating the prediction modes of 4×4 luminance blocks; Yang et al. [12] also leverage the intra prediction modes.

All the aforementioned works are based on either MPEG-x or H.26x standards, which do not fit to the latest highly-efficient HEVC standard. On the other hand, when employing HEVC there appear less articles in literature. In [5], authors propose a robust video steganography scheme for intra and inter frames by manipulating the AC coefficients. In [13], authors propose to also study the AC coefficients to embed every watermark bit in 4×4 intra blocks. In [14] the watermark information is embedded within the quantized DCT coefficients as well. Chang et al. [15] employ a DCT and Discrete Sine Transform (DST) based coefficient perturbation scheme for embedding bits in HEVC without inducing intra-frame error propagation. Wang et al. [16] presented a large-capacity information hiding approach for HEVC Video using intra-prediction mode. Authors in [17] proposed a video steganography algorithm without bit-rate increase, where information is embedded in the stage of entropy coding by exploiting the new features of CABAC for HEVC.

The main drawback of the presented approaches is that they all require to modify the standard, which makes it difficult to deploy them into commercial systems. Therefore, in this paper we propose a method for hiding information in videos that are coded with the HEVC, but without modifying the standard and with a negligible quality loss. Moreover, a method for recovering such information is proposed as well. Several high-resolution sequences will be employed for testing the proposed methods.

The steganalysis tools face four extremely complex problems: determining if there is information hidden in the carrier, estimate the size of this information, locate its exact location and, finally, extract it. For these purposes, there are several types of steganographic technique attacks, which can be listed as visual attacks, structural attacks and statistical attacks. The first one references to the analysis of the image and verifying the differences between the original and the manipulated frames. The second type tries to guess the properties of the algorithms used to hide the information. Finally, the last subset of attacks performs statistical calculations to try detecting if there is hidden information; among them we can find the chi-square attack, developed by Westfield and Pfitzmann. These are much more efficient than the previous ones [18].

Most known steganalysis tools try to find hidden messages using any of the above techniques on images. To the best of our knowledge, there is no available tool for videos. Some of these tools focus on detecting hidden information by LSB methods, as OpenStego, StegExpose and Xiao. Others as VSL and StegSecret cover a higher spectrum. For instance, they also cover RS attacks, which are very efficient for detecting the presence of a message in a digital image [19]. The work in [19] also provides tools to detect visual alterations in the frames and supplies tools to perform visual attacks and ChiSquare attacks.

3. HEVC preliminaries

In this section we briefly describe some multimedia and HEVC details that will be important to understand the proposal.

Videos are composed of a sequence of images, also known as frames, which are constituted by a set of basic units called pixels. These pixels define the color and can be represented in many formats, as RGB [2]. In this paper, the sequences employed for testing the proposed framework are constructed using the YUV420 format. Any YUV-like representation is built around three values: the luminance (Y) and the color tone or chrominance (U and V, aka Cb and Cr), each component being defined by 8-bits. The value appended at the end indicates the proportion of samples associated to the format under consideration. In our case, YUV420 means that for every luminance sample, the chrominance is subsampled by a horizontal factor of 2 and by a vertical factor of 2 as well. These facts are illustrated in Fig. 1.

It should be noted that there is a direct mathematical relationship between the YUV planar format and RGB color space, as shown in Fig. 2. Sometimes it is needed to make conversions between formats in order to analyze certain features of the video frames.

The HEVC encoding process is based on Coding Tree Units (CTUs), being each CTU composed of several Coding Units (CUs) [2]. Considering the YUV420 format, this means that every CU is composed of a coding block of luminance and two coding blocks of chrominance subsampled as previously mentioned. Fig. 3 shows this. The maximum size of the CU is 64×64 pixels and the smallest is 8 × 8 pixels. The Prediction Unit (PU) is a division of the CU, containing one luminance sample and two chrominance samples and it is used to forecast the samples of the prediction block. The maximum dimension is the size of the associated CU and the minimum dimension is 4 × 4 pixels.

In order to encode the video, the HEVC standard can apply two compression modes, independently or jointly. The first one, known as intra-frame compression and of higher quality, refers to a spatial model, where each frame is compressed separately. The second one is the inter-frame compression, which is a temporal compression, using similarities between past and future frames. The inter-frame compression system is not random; in order to determine the relationships between past and future frames it is necessary to specify the order in which the images are treated. This provision is defined in the Group of Pictures (GOP).

It should be noted that H.265 is a lossy compression process where a quantization parameter (QP) is used to compress a range of values to a single value. Therefore, the higher the quantization, the larger the loss of quality. To correct the distortion between the original and reconstructed samples, the Sample Adaptive Offset (SAO) filter is frequently applied.

The result of the coding process is a HEVC bitstream that consists of an orderly sequence of syntactic elements. The key components consist of a sequence of data units called Network Abstraction Layers (NALs). Some of these NALs contain high-level information related to the entire encoded video sequence or to a subset of images within it (fps, size, aspect ratio). The Slice NAL units are a subset of NALs that carry coded samples. These Slice NAL units contain data from the frames of the

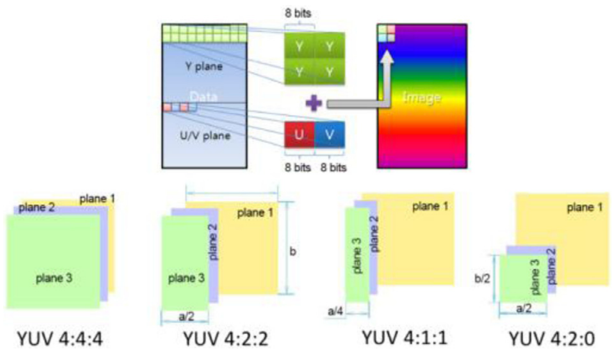


Fig. 1. YUV format.

$$\begin{bmatrix} R \\ G \\ B \end{bmatrix} = \begin{bmatrix} 1.000 & 0.000 & 1.140 \\ 1.000 & -0.395 & -0.581 \\ 1.000 & 2.032 & 0.000 \end{bmatrix} \cdot \begin{bmatrix} Y \\ U \\ V \end{bmatrix} \quad \begin{matrix} R \in [0, 255] \\ G \in [0, 255] \\ B \in [0, 255] \end{matrix}$$

Fig. 2. Conversion between RGB and YUV.

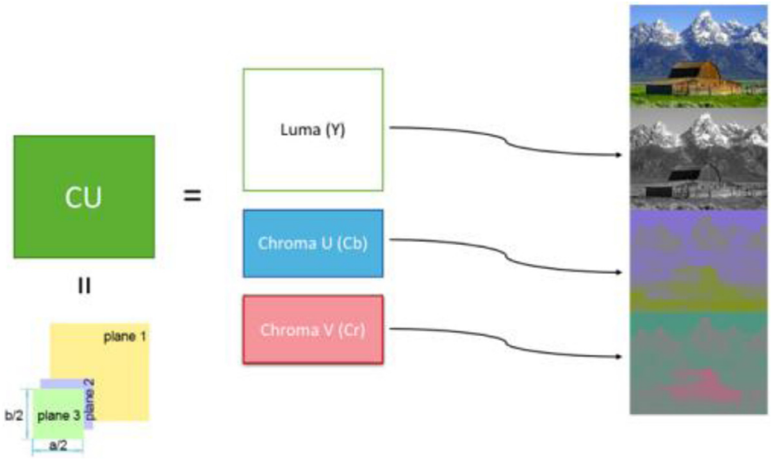


Fig. 3. CU with YUV420 format.

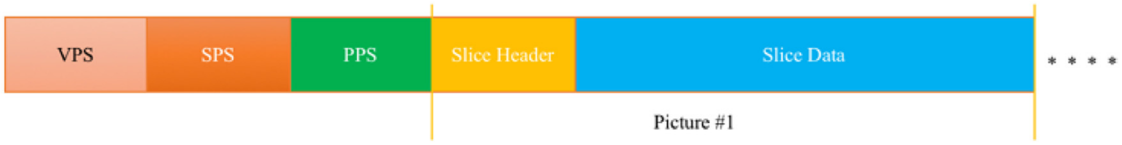


Fig. 4. HEVC bitstream NAL structure.

encoded video. They can contain a complete frame or part of it and each one of them can be decoded independently; that is, without using information from another slice. VPS (Video Parameter Set), SPS (Sequence Parameter Set) and PPS (Picture Parameter Set) contain general video parameters. They provide a robust mechanism for carrying data that are essential in the decoding process. Note that they can be part of the bitstream or can be stored separately. Fig. 4 shows a structural diagram of the HEVC bitstream.

4. Embedding information

In this section our proposed method for hiding information is described in detail.

As shown in Fig. 5, our proposal consists of inserting every bit of information, either plain or ciphered, in random 4 × 4 intra-blocks. The selected block then shall be named Insertion Block (IB) hereafter. The positions of these IBs are randomly chosen, which increases the strength to face possible attacks, and fully defined by the tuple (frame, X, Y). Thus, the number of required tuples will be equal to the number of blocks to be encoded. It must be noted that we only need to know the location of the leftmost and topmost pixel of each block, since the rest pixels within the block are always in a 4 × 4 region. The X and Y coordinates at the beginning of the affected block must be a multiple of 4 to ensure that the block can be a sub-block of a CU or coincide with a PU.

Although luminance is more sensitive to the human eye than chrominance [20], as shown in Fig. 3 in YUV420 format there are four luminance samples for each color sample, so a slight change on the Y plane will be less distorting than a change in the U or the V planes.

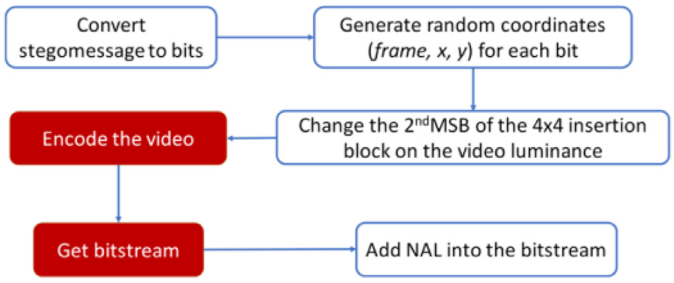

Convert stegomessage to bits
Generate random coordinates (frame, x, y) for each bit
Change the 2nd MSB of the 4x4 insertion block on the video luminance
Encode the video
Get bitstream
Add NAL into the bitstream

Fig. 5. Hiding information flow.

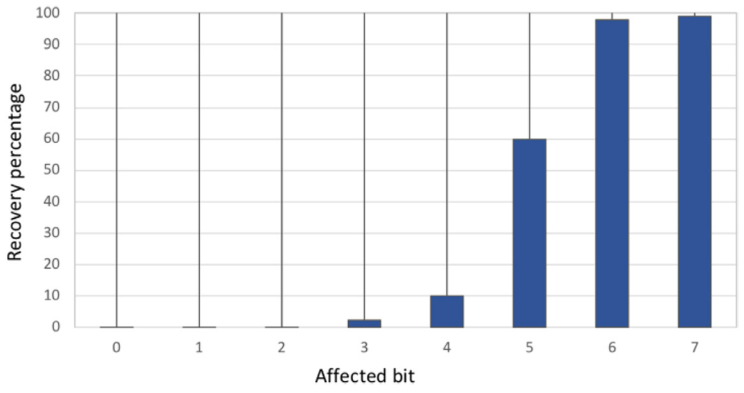


Fig. 6. Average recovery percentage according to the bit position (position 0 corresponds to the LSB and position 7 to the MSB). Results were obtained after 50 different encodings.

The hiding process transforms the message to be hidden into an array of bits. Each of these bits will be the second most significant bit (MSB) of the IB located in a random frame and coordinates. The reason for selecting this bit is drawn from the study depicted in Fig. 6. A total of 50 different tests have been performed with a set of videos with different resolutions from [21]. These video sequences have been normalized to YUV420 format and a bit depth of 8 bits, and are distinct from the videos used to experimentally validate the proposed steganographic algorithm. On average we realized that modifying a bit from positions 0 to 4 lead to being unable to recover almost anything. Bit 5 achieved 60% recovery percentage and bit 6 $>$ 95%. Hence, bit 6 is a good candidate for guaranteeing the recovery. The drawback is that this might cause some undesired effects to appear if the area on which it is applied is homogeneous, as bit 6 has an important weight within an 8-bit value. Nevertheless, from the Quality of Experience (QoE) point of view this may become even imperceptible because of the frame rate. Although the human eye does not work in terms of frames per second, a frame ratio close to 30 fps, produces smooth motion and is similar to how we see the world. A frame ratio higher than this is used in scenes with a lot of motion or slow motion, but under these conditions most modifications go unnoticed too because of the speed while passing the frames.

Hence, the current proposal needs to employ a 4×4 intra-block per hidden bit. Thus, as every block is composed of 16-bits, and every character occupies 8-bits, a total amount of $16 \times 8xL$ pixels will be modified, being L the length of the message to embed.

One of the advantages of this method lies in the fact that the insertion of information is done in a completely random way. But on the other hand, this implies the receiver knowing the insertion positions to recover the information. However, the aforementioned tuples (cyphered or not) that define the positions of the IBs can be attached in a new NAL of the bitstream that, in turn, can be transmitted separately to the bitstream containing the video itself. This NAL shall be named Coordinate Parameter Set (CPS) and it possesses the standard FD_NUT (0×26) data type, which can be recognized by all the HEVC decoders.

In HEVC, unlike other systems, there is a protocol to align and separate NAL units. At the beginning of each NAL, a three or four byte code ($0 \times 000{,}001$ or $0 \times 00{,}000{,}001$) is added. We will add the sequence $0 \times 00{,}000{,}001$ to the start of our new CPS NAL. Next, we apply the syntax for the header of a NAL unit, following the code specified in Table 1.

The binary value after attaching the aforementioned code is [0,100,110,000,000,001]b, or $0 \times$ 4c01 in hexadecimal. Finally, we will attach the emulation prevention bytes, whose value is $0 \times 00{,}000{,}301$. When this value is in the NAL unit, it must be discarded during the decoding process. Thus, the complete header of our NAL unit will be: $0 \times$ 000000014c0100000301, with a size of 10 bytes.

After the header, the tuples that define the location of the IBs will be included. An amount of 6 bytes is enough to define each of them, allocating two bytes for the frame (although there could be more bytes in case the number of frames exceeds

Table 1
NAL unit header syntax (Rec. ITU-T H.265 v4 (12/2016)).

Code	Size in bits	Value
nal_unit_header() {	–	–
forbidden_zero_bit	1	0
nal_unit_type	6	100110b = 0×26
nuh_layer_id	6	0
nuh_temporal_id_plus1	3	1
}	–	–

65,535 units), two bytes for the X coordinate and the last two bytes for the Y coordinate. The insertion has been done in little endian, following the format in which the data is stored in Intel processors. Lastly, in order to make the extraction of the tuples easier, a pattern with value 0xffff is appended to the CPS NAL tail. This hexadecimal value does not coincide with any of the NAL unit type codes defined in the H.265 standard, so it can not be confused by an original Network Abstraction Layer of the bitstream. In this way we have delimited the location of the tuples, since they are completely defined between the header of the CPS NAL and the defined closure pattern.

Although the results of the recovery process are positive (especially with low QP values), in some cases the recovery of the whole message is not assured. Optionally, in order to guarantee a complete recovery, the message to be hidden can be added by triplicate, following the algorithm shown above. This has the disadvantage of increasing by three times the size of the CPS NAL, but empirically it has been shown that it guarantees the complete recovery of the message and decreases the probability of an attacker extracting the information, since the number of embedded characters noticeably augments, as will be commented in Section 7.5.

4.1. Illustrative example

In Fig. 7 there is an example of the proposed technique. The term “war” is being concealed as an example, whose ASCII representation is 0 $\times$ 776,172 or, in binary, [01,110,111 01,100,001 01,110,010]b. Then, each of these bits will become the value of the second most significant bit within the 16 pixels of the IB. Thus, 384 pixels will be modified overall. As observed, in this case the quality of the frames is maintained.

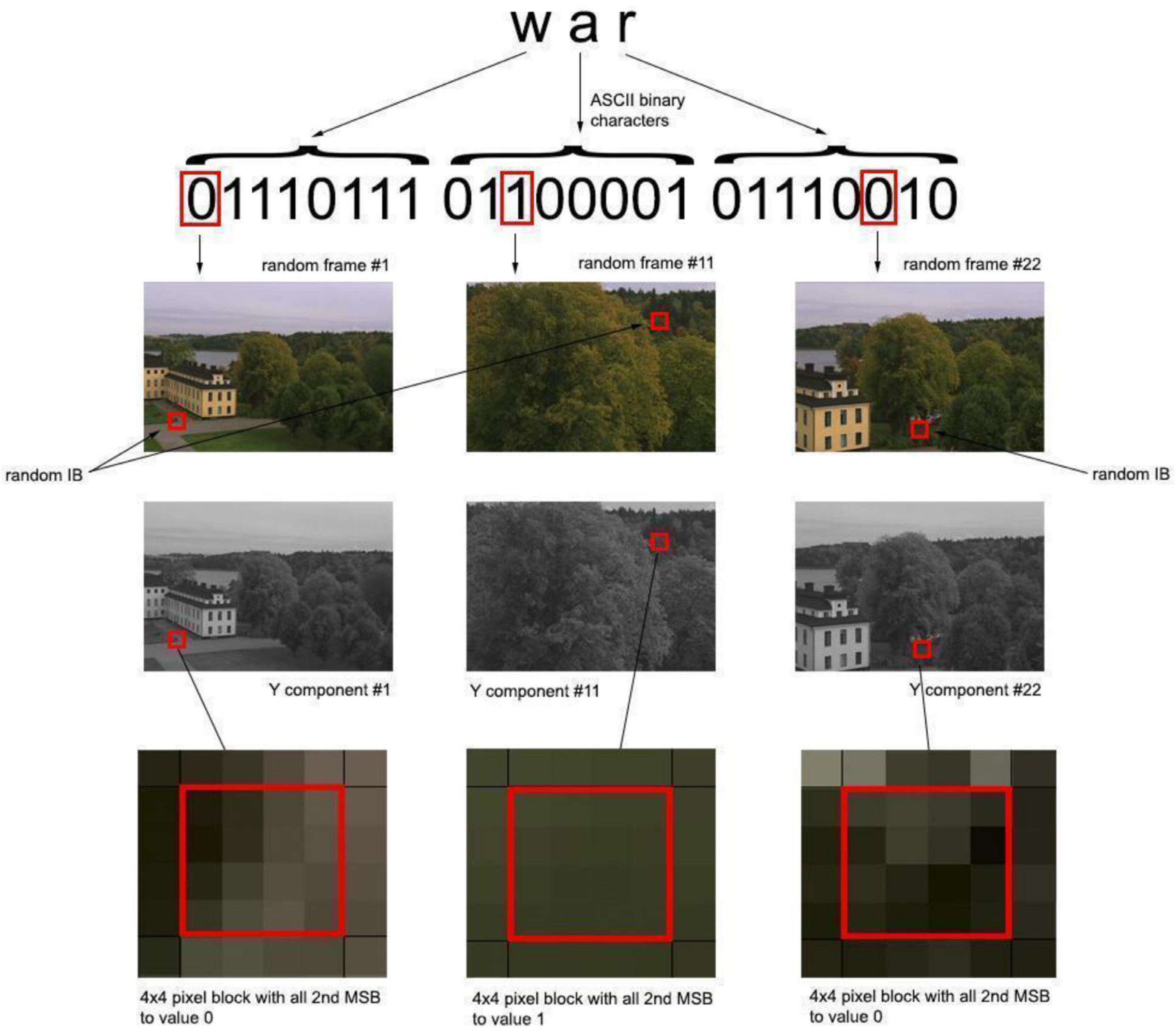


Fig. 7. Hiding ‘war’ within a sequence.

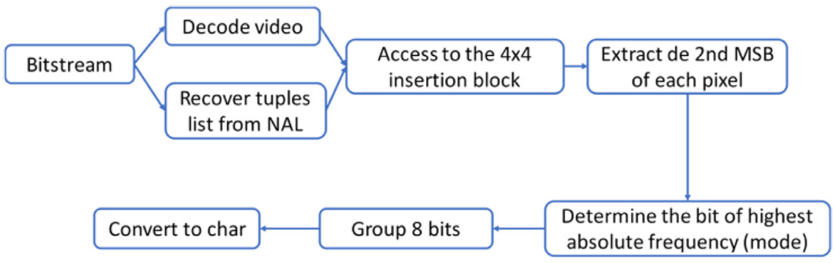

Bitstream
Decode video
Recover tuples list from NAL
Access to the 4x4 insertion block
Extract de 2nd MSB of each pixel
Determine the bit of highest absolute frequency (mode)
Group 8 bits
Convert to char

Fig. 8. Retrieving information flow.

Table 2
Triple redundancy to guarantee message integrity.

Orginal message	Encoded message	Recovered message with error control
I need a coffee!	I n3ed a coffee!	I need a coffee!
I need a coffee!	I need a coffe!!	
I need a coffee!	I needa Coffee!	

Finally, in order to show how the NALs are constructed, let us consider a couple of IBs identified by the tuples (frame, X, Y) with values (0 × 16d, 0 × 189, 0xd) and (0 × 1d0, 0 × 34d, 0 × 71). Then, the bytes inserted in little endian will be 0 × 6d0189010d00 and 0xd0014d037100 for the first and second tuple, respectively.

5. Retrieving information

In this section, the method for recovering the information is described in detail. The flow for doing this is illustrated in Fig. 8.

The first thing to note is that both the video and the tuples contained in the NALs are necessary to locate the information. Once the video is decoded, it is possible to access the proper frame and position to collect the second MSB of the 16 luminance pixels integrating the IB. After computing the mode among these 16 bits, it is possible to interpret the bit that was inserted. In order to fully identify the ASCII code corresponding to the character previously inserted, it is necessary to group 8 recovered bits. By repeating this process, it is possible to retrieve the whole embedded message.

If the triple redundancy is employed to guarantee the integrity of the message, the extraction process does not change. Simply, at the time of composing the message the algorithm has to consider the most repeated character in every position of the message, as can be seen in Table 2.

5.1. Illustrative example

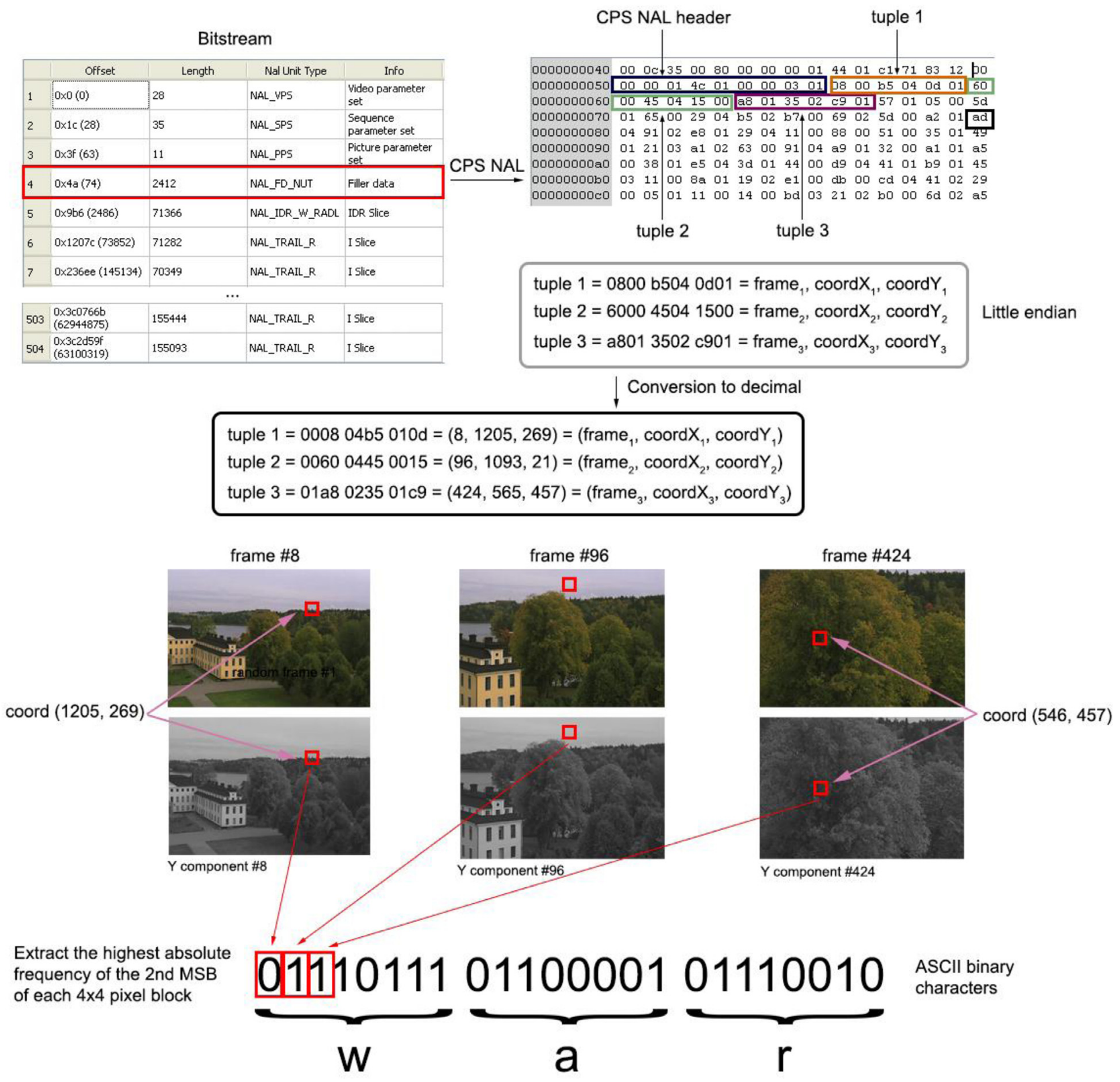
In Fig. 9 there is an example of the proposed technique. Once the bitstream is received, the CPS NAL header [0 × 000000014c0100000301] and its tail [0xffff] must be extracted. It must be reminded that each one of the NAL can be sent as an annex to the rest of data or separately. The information must be grouped by tuples, from 6 to 6 bytes sequentially [0 × 0800b5040d01; 0 × 600,045,041,500; 0xa8013502c901, ...]. It is very important to keep them ordered since this will give meaning to the recovered message.

Each group of 6 bytes is in little endian format, being subdivided into groups of 2 bytes, which composes the tuple (frame, X, Y). Next, each component in little endian is interpreted and converted to into decimal values [(0 × 0008, 0 × 04b5, 0 × 010d); (0 × 0060, 0 × 0445, 0 × 0015); (0 × 01a8, 0 × 0235, 0 × 01c9), ...] = [(8, 1205, 269), (96, 1093, 21), (424, 565, 457), ...]. Once the tuple is composed, the recovery algorithm accesses the corresponding luminance plane of the selected frame. The X and Y coordinates then represent the beginning of an insertion block (IB) formed by a quadrant of 4 × 4 pixels. From each 4 × 4 block, the second most significant bit must be extracted and the value with the highest absolute frequency will be selected. After extracting 8 bits, an ASCII character can be constructed. Lastly, in this example only the first three bits of the character ’w’ are extracted. By repeating this process, it is possible to retrieve the original term ‘war’, inserted in Section 4.1.

6. Metrics and framework

The metrics as well as the framework employed to evaluate the proposal are presented in this section.

The Structural Similarity Index (SSIM) is a quality metric to measure the similarity between two images. The SSIM can be seen as a measure of the quality of one of the images being compared, provided that the other image is considered to possess a perfect quality. Similarly, the Peak Signal-to-Noise Ratio (PSNR) measures the relationship between noise and



	Offset	Length	Nal Unit Type	Info
1	0x0 (0)	28	NAL_VPS	Video parameter set
2	0x1c (28)	35	NAL_SPS	Sequence parameter set
3	0x3f (63)	11	NAL_PPS	Picture parameter set
4	0x4a (74)	2412	NAL_FD_NUT	Filler data
5	0x9b6 (2486)	71366	NAL_IDR_W_RADL	IDR Slice
6	0x1207c (73852)	71282	NAL_TRAIL_R	I Slice
7	0x236ee (145134)	70349	NAL_TRAIL_R	I Slice
	...			
503	0x3c0766b (62944875)	155444	NAL_TRAIL_R	I Slice
504	0x3c2d59f (63100319)	155093	NAL_TRAIL_R	I Slice

```
0000000040  00 0c 35 00 80 00 00 00 01 44 01 c1 71 83 12 00
0000000050  00 00 01 4c 01 00 00 03 01 08 00 b5 04 0d 01 60
0000000060  00 45 04 15 00 a8 01 35 02 c9 01 57 01 05 00 5d
0000000070  01 65 00 29 04 b5 02 b7 00 69 02 5d 00 a2 01 ad
0000000080  04 91 02 e8 01 29 04 11 00 88 00 51 00 35 01 49
0000000090  01 21 03 a1 02 63 00 91 04 a9 01 32 00 a1 01 a5
00000000a0  00 38 01 e5 04 3d 01 44 00 d9 04 41 01 b9 01 45
00000000b0  03 11 00 8a 01 19 02 e1 00 db 00 cd 04 41 02 29
00000000c0  00 05 01 11 00 14 00 bd 03 21 02 b0 00 6d 02 a5
```

tuple 1 = 0800 b504 0d01 = frame_1, coordX_1, coordY_1
tuple 2 = 6000 4504 1500 = frame_2, coordX_2, coordY_2
tuple 3 = a801 3502 c901 = frame_3, coordX_3, coordY_3

tuple 1 = 0008 04b5 010d = (8, 1205, 269) = (frame_1, coordX_1, coordY_1)
tuple 2 = 0060 0445 0015 = (96, 1093, 21) = (frame_2, coordX_2, coordY_2)
tuple 3 = 01a8 0235 01c9 = (424, 565, 457) = (frame_3, coordX_3, coordY_3)

01110111 01100001 01110010

Fig. 9. Retrieving 'war' within a bitstream.

signal in decibels. It is commonly used as a quality measure between the original and a compressed image; the higher PSNR value, the better quality has the compressed image.

A random stego-message consisting of 50 printable and non-printable characters (such as spaces and carriage returns) has been considered for insertion. 30 different sequences with high resolutions as FullHD and 4k have been tested, utilizing YUV 4:2:0 8-bit format, 500 frames with a refresh rate of 50 fps for 720 i/p frame dimensions (HD), 1080i/p (FullHD) [21-22] and 300 frames with a refresh rate of 30 fps for 2160i/p (4k) [23-25]. The encoding is performed using the HM-16.2 reference software and x265, in order to test with a real-time encoder. The configuration parameters for both encoders are shown in Table 3.

7. Experiments

This section presents and comments our results referred to the bitstream sizes, the recovery capacity employing different QP values and encoder types. Finally, a quality evaluation of the videos using both HM-16.2 and x265 will be shown.

Table 3
Configuration parameters for HM-16.2 and x265 encoders.

Profile	Main
Bit depth	8
Chroma subsampling	4:2:0
Maximum CU size (width and height)	64
Analysis of asymmetric motion partitions	Enable
Max intra period in frames	Forced all-intra
GOP size	1 (forced all-intra)
Motion search range	64
SAO	Enabled

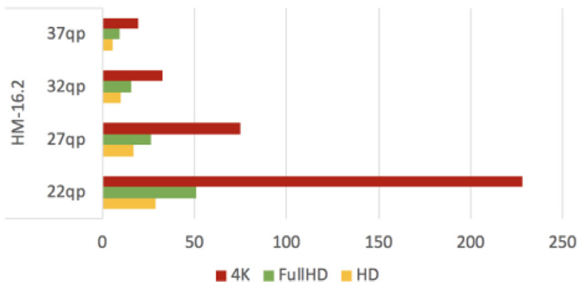


Fig. 10. Average bitstream size (MB). Resolution vs. QP with HM-16.2 encoder.

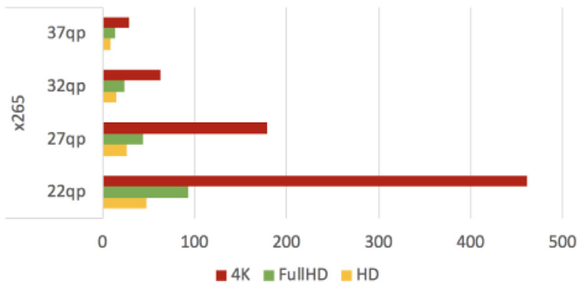


Fig. 11. Average bitstream size (MB). Resolution vs. QP with x265 encoder.

7.1. Bitstream size comparison per encoder

According to the specifications proposed and recommended by the JCT-VC, the algorithm proposed in this paper has been validated for the quantization parameters (QP) 22, 27, 32 and 37 for the encoders HM-16.2 (single-threaded) and x265 (multi-threaded). All the videos [21-25] have been encoded for each of the following resolutions: HD (1280 × 720), FullHD (1920 × 1080) and 4k (3840 × 2160).

Figs. 10 and 11 confirm that the size of the bitstream resulting from the coding process has a small variation depending on the employed encoder. A higher compression rate is perceived in HM-16.2 in comparison with x265 when using the same configuration parameters. Nevertheless, HM-16.2 takes many hours to achieve the desired result, while x265 just requires some minutes, which is one of its objectives to reach real time video coding and transmission. Table 4 shows the coding time of video sequences with and without stego-message with HM-16.2 and x265 encoders. It also shows the time to hide the message and to add the new CPS NAL, where both tasks are independent from the codification process. Table 4 also highlights that hiding information and adding a new NAL does not affect the codification process and it is negligible in comparison with the encoding time.

A bitstream of smaller size considerably improves the transmission time of data in a communication network, but the quality of the resulting video gets worse, as the coding process is an attack in itself, as will be seen in the following subsections when studying the quality achieved by both encoders. This fact directly affects the stego-message that we can include in our video carrier, being able to suffer losses in which it is impossible to find out exactly what the original message was, although these losses can be mitigated by utilizing triple redundancy.

Table 4
Time comparative between encoding process vs. the proposed steganographic method.

		HM-16.2 encoder		X265 encoder		
	Video	Without stego-message	With stego-message	Without stego-message	With stego-message	Aggregation time (stego-message + CPS NAL)
HD	crowd_run.yuv	51 min 24s	51 min 37s	32s	31s	<2s
	ducks_take_off.yuv	51 min 56s	51 min 33s	31s	32s	<2s
	in_to_tree.yuv	53 min 32s	53 min 40s	32s	32s	<2s
	old_town_cross.yuv	52 min 27s	52 min 15s	33s	32s	<2s
	park_joy.yuv	51 min 44s	51 min 42s	32s	33s	<2s
	green_land.yuv	53 min 8s	53 min 28s	33s	33s	<2s
	promenade.yuv	53 min 54s	54 min 9s	32s	33s	<2s
	river.yuv	53 min 50s	54 min 16s	34s	33s	<2s
	street_lights.yuv	51 min 41s	51 min 52s	31s	30s	<2s
	sunny_day.yuv	51 min 57s	52 min 10s	30s	30s	<2s
FullHD	crowd_run.yuv	1 h 52 min 23s	1 h 52 min 20s	1 min 5s	1 min 6s	<2s
	ducks_take_off.yuv	1 h 54 min 48s	1 h 54 min 53s	1 min 3s	1 min 3s	<2s
	in_to_tree.yuv	1 h 53 min 9s	1 h 53 min 4s	1 min 8s	1 min 8s	<2s
	old_town_cross.yuv	1 h 54 min 37s	1 h 55min	1 min 10s	1 min 9s	<2s
	park_joy.yuv	1 h 56 min 15s	1 h 53 min 5s	1 min 2s	1 min 2s	<2s
	green_land.yuv	1 h 55 min 16s	1 h 56 min 46s	1 min 7s	1 min 7s	<2s
	promenade.yuv	1 h 55 min 23s	1 h 55 min 27s	1 min 6s	1 min 6s	<2s
	river.yuv	1 h 56 min 17s	1 h 57 min	1 min 9s	1 min 9s	<2s
	street_lights.yuv	1 h 51 min 42s	1 h 51 min 53s	1 min 2s	1 min 2s	<2s
	sunny_day.yuv	1 h 52 min 24s	1 h 53 min 26s	1 min 1s	1 min 1s	<2s
4k	bund_night_scape.yuv	4 h 27 min 14s	4 h 27 min 58s	2 min 27s	2 min 27s	<2s
	campfire_party.yuv	4 h 5 min 56s	4 h 6 min 2s	2 min 36s	2 min 35s	<2s
	rush_hour.yuv	4 h 45 min 3s	4 h 44 min 57s	2 min 22s	2 min 21s	<2s
	tall_buildings.yuv	4 h 32 min 52s	4 h 32 min 58s	2 min 13s	2 min 14s	<2s
	tree_shade.yuv	4 h 3 min 40s	4 h 3 min 30s	2 min 49s	2 min 51s	<2s
	basketball.yuv	4 h 2 min 58s	4 h 3 min 35s	2 min 12s	2 min 10s	<2s
	construction.yuv	4 h 38 min 52s	4 h 37 min 19s	2 min 36s	2 min 37s	<2s
	flowers.yuv	4 h 11 min 51s	4 h 9 min 59s	2 min 15s	2 min 15s	<2s
	lake.yuv	5 h 4 min 13s	5 h 5 min 30s	2 min 58s	2 min 56s	<2s
	park.yuv	4 h 33 min 40s	4 h 31 min 0s	2 min 31s	2 min 31s	<2s

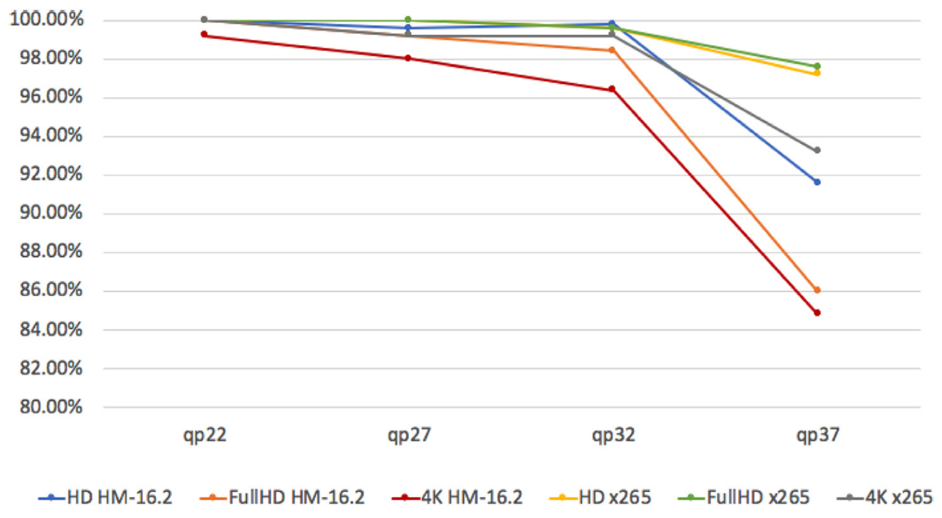


Fig. 12. Average recovery rate per encoder and QP.

7.2. Recovery capacity per QP and encoder

In Fig. 12, the percentage of recovered characters is shown. It is important to warn that it is a recovery process which has not applied any control error. That is, the analyzed result is the raw message extracted directly from the compressor. It can be seen that the recovery capacity decreases as the quantification parameter increases. This fact happens in both encoders, being more noticeable for HM-16.2.

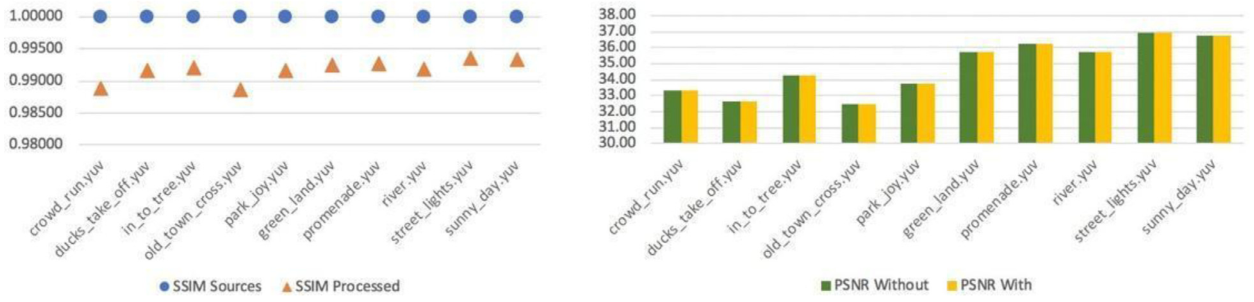


Fig. 13. SSIM and PSNR evaluation of FullHD sequences. Encoding performed with HM-16.2.

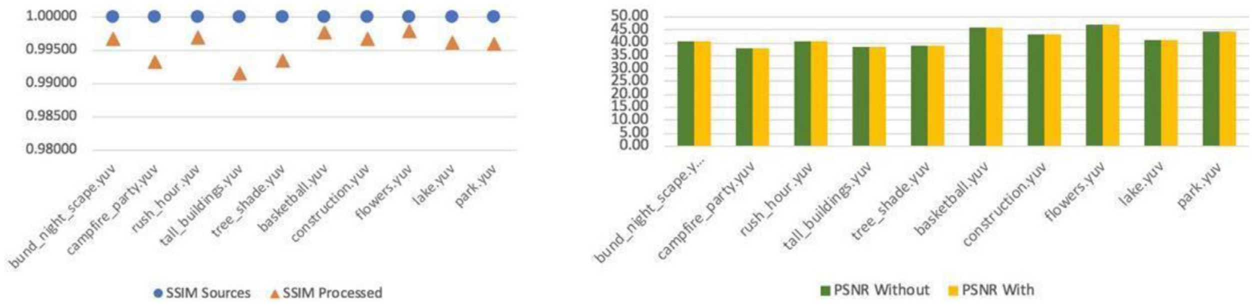


Fig. 14. SSIM and PSNR evaluation of 4k sequences. Encoding performed with HM-16.2.

Performing the compressions with QP = 22, the recovered message is optimal for both encoders (except for HM-16.2 in 4k, which retrieves slightly more than 99% of the added characters). While it is true that in the worst case in our experiments (see 4k resolution, encoder HM-16.2 and QP = 37 in Fig. 12), the message may not be detected correctly, when the triple redundancy is applied the message can be completely retrieved.

7.3. Quality evaluation on HM-16.2

In this experiment, the quality of the resulting videos has been evaluated. For this purpose, the settings shown in Table 3 have been employed on the HM-16.2 reference software for FullHD and 4k, taking QP = 32 for FullHD and QP = 27 for 4k resolution.

Fig. 13 contains the average SSIM and PSNR results for ten FullHD sequences, respectively. The chart on the left depicts two subsets of results. Those labelled as *SSIM Sources* show the SSIM difference between the original video and the original video after embedding the message, while those labelled as *SSIM Processed* represent the SSIM difference between the compressed videos after decoding without and with the stego-message. As can be observed, the SSIM values of the processed version are very close or even higher than 0.99, which demonstrates that the structural similarity is maintained after embedding the message, so that it is not easy to determine whether or not there are some hidden data. The chart on the right shows the PSNR of two subsets of results as well: on the one hand those labelled as *PSNR Without* show the PSNR difference between the original video and their decoded version and, on the other hand, those labelled as *PSNR With* depict the PSNR difference between the video containing the stego-message prior to encoding and after decoding. As observed, the PSNR is almost identical between the two subsets of results for all the sequences, which confirms the aforementioned similarity thesis.

In order to complete the study, several 4k sequences have been encoded with HM-16.2. Fig. 14 contains an SSIM and a PSNR comparison for the 4k videos, as in the case of FullHD. As it is shown, the similarity even increases. This fact is logical, since a 4k frame is four times a FullHD one. In other words, hiding a message is easier since there are more pixels. Nevertheless, it must be mentioned that the HEVC encoder becomes more aggressive on 4k frames, especially with high values of the quantization parameter.

7.4. Quality evaluation on x265

In this experiment we have tested the same FullHD and 4k sequences but using x265 in order to employ a real-time encoder. The QP value is 32 for FullHD sequences and 27 for 4k, as in the previous experiment.

Figs. 15 and 16 show the SSIM metrics and the PSNR results for both FullHD and 4k sequences, respectively. On the left of Fig. 15 it is possible to observe how the SSIM of the processed videos is around 0.995, while in the chart on the right the PSNR loss is imperceptible when embedding the stego-message. When considering 4k, Fig. 16 shows a SSIM that sometimes

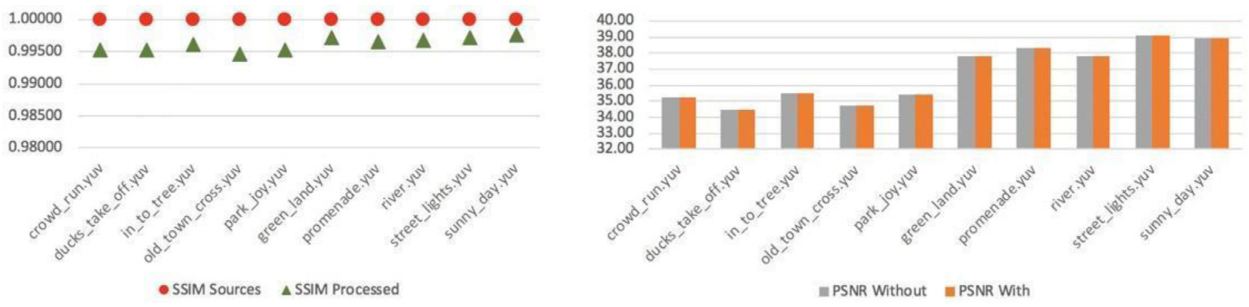


Fig. 15. SSIM and PSNR evaluation for FullHD sequences. Encoding is performed with x265.

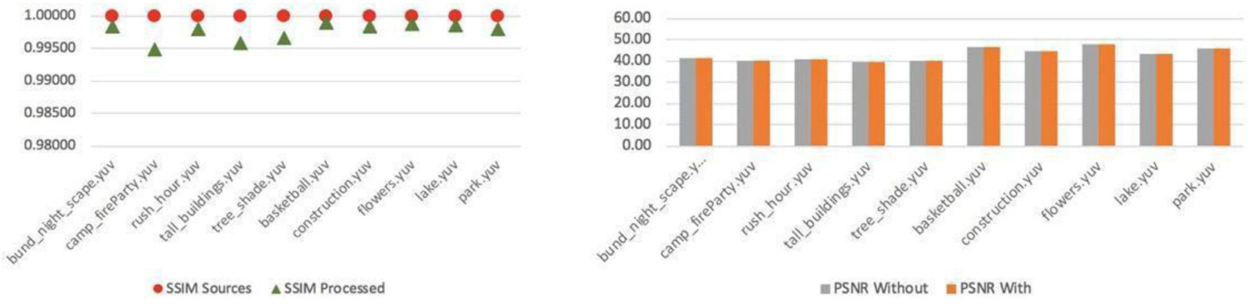


Fig. 16. SSIM and PSNR evaluation for 4k sequences. Encoding is performed with x265.

is even superior to that for FullHD. However, the PSNR remains almost equal. Finally, it is interesting to note that the quality metrics depicted in Figs. 13–16 show how the x265 real-time encoder behaves better than HM-16.2, being able to obtain videos with higher quality.

7.5. Validation of the algorithm with steganalysis tools

In this experiment we have used several representative steganalysis tools with videos that carry on stego-messages and coded with x265: OpenStego, VSL, Xiao Steganography, StegSecret and StegExpose. Nevertheless, employing them is not straightforward. Generally, attacks are based on particularities of images taken independently of the rest of the composition of the video, so we have been forced to transform each frame into images in jpg, png or bmp formats, depending on the tool. In the same way, some of these tools are only able to detect hidden information if the employed steganographic algorithm has been the one proposed by the same tool.

OpenStego, VSL and Xiao Steganography have been unable to find any embedded information, but StegSecret and StegExpose have detected some frames containing modified pixels. Tables 5 and 6 show these results for HD, FullHD and 4k resolutions. The column labelled as 'Modified Frames' shows the number of frames in which the proposed algorithm has included hidden information out of the 500 frames (for HD and FullHD) or 300 frames (for 4k) that compose each video, while the column labelled as '% detected frames' indicates, per image format (bmp, jpg and png), the percentage of false positives (frames detected as carriers that are not), percentage of frames correctly detected but the information size is wrong, and the percentage of frames correctly detected. The last three results have been calculated with respect to the total number of frames that compose the video and are separated by a slash.

False negatives are the difference between the value of the column "modified frames" and the frames correctly detected, even if the size of the information is not adequate.

In Table 5, an evaluation of StegSecret is shown when employing input frames converted to bmp and jpg formats. It should be noted that the tool efficiency using bmp is greater than employing jpg. In Table 6, the evaluation of StegExpose when utilizing input frames converted to bmp and png formats is displayed. In this case, the efficiency for bmp and png is exactly the same.

In any case, none of the aforementioned tools has been able to find the hidden information. It should be noted that, even developing a perfect steganoanalytic tool that detects each of the insertion blocks (IB) and finds out without any type of error the correct bit, it must be able to sort them in such a way that it could extract the desired character. If the hidden message is composed by 50 characters (bit depth=8), sorting the 400 bits in such a way as to ensure that the correct message is found implies a variation without repetition of 400 elements taken in groups of 8 bits; i.e. 400!/392! different options. Even computationally with current technology, it is unaffordable to calculate all the different options in a prudential time.

Table 5
StegSecret steganalysis results for frames converted to bmp and jpg.

	Video with stegomessage	Modified frames	% detected frames (bmp)	% detected frames (jpg)
HD	crowd_run.yuv	273	0/0/0	0/0/0
	ducks_take_off.yuv	271	0/0/0	0/0/0
	in_to_tree.yuv	280	0/0.2/0	0/0/0
	old_town_cross.yuv	277	0/0/0	0/0/0
	park_joy.yuv	279	0/0/0	0/0/0
	green_land.yuv	283	0/0/0	0/0/0
	promenade.yuv	270	0.2/0.8/0	0/0/0
	river.yuv	276	0/0/0	0/0/0
	street_lights.yuv	274	0/0/0	0/0/0
	sunny_day.yuv	279	0/0/0	0/0/0
FullHD	crowd_run.yuv	279	0.2/0/0	0/0/0
	ducks_take_off.yuv	275	0.8/0.4/0	0/0/0
	in_to_tree.yuv	268	0.4/0.4/0	0/0/0
	old_town_cross.yuv	280	0/0/0	0/0/0
	park_joy.yuv	271	0/0/0	0/0/0
	green_land.yuv	270	0/0/0	0/0/0
	promenade.yuv	277	0/0/0	0/0/0
	river.yuv	265	0/0/0	0/0/0
	street_lights.yuv	279	2.4/3/2000	0/0/0
	sunny_day.yuv	268	0.2/0.2/0	0/0/0
4k	bund_night_scape.yuv	216	0/0/0	0/0/0
	campfire_party.yuv	216	0/0/0	0/0/0
	rush_hour.yuv	216	0.7/0/0	0/0/0
	tall_buildings.yuv	222	0/0/0	0/0/0
	tree_shade.yuv	225	0/0/0	0/0/0
	basketball.yuv	220	0/0/0	0/0/0
	construction.yuv	231	0/0/0	0/0/0
	flowers.yuv	227	0/0.3//0	0/0/0
	lake.yuv	219	0/0/0	0/0/0
	park.yuv	214	1/2.3/0	0/0/0

Table 6
StegExpose steganalysis results for frames converted to bmp and png.

	Video with stegomessage	Modified frames	% detected frames (bmp)	% detected frames (jpg)
HD	crowd_run.yuv	273	0/0/0	0/0/0
	ducks_take_off.yuv	271	0/0/0	0/0/0
	in_to_tree.yuv	280	0/0/0	0/0/0
	old_town_cross.yuv	277	0/0/0	0/0/0
	park_joy.yuv	279	0/0/0	0/0/0
	green_land.yuv	283	0/0/0	0/0/0
	promenade.yuv	270	0/0/0	0/0/0
	river.yuv	276	0/0/0	0/0/0
	street_lights.yuv	274	0/0/0	0/0/0
	sunny_day.yuv	279	0/0/0	0/0/0
FullHD	crowd_run.yuv	279	0/0/0	0/0/0
	ducks_take_off.yuv	275	0/0/0	0/0/0
	in_to_tree.yuv	268	0/0/0	0/0/0
	old_town_cross.yuv	280	0/0/0	0/0/0
	park_joy.yuv	271	0/0/0	0/0/0
	green_land.yuv	270	0/0/0	0/0/0
	promenade.yuv	277	0/0/0	0/0/0
	river.yuv	265	0/0/0	0/0/0
	street_lights.yuv	279	0/0/0	0/0/0
	sunny_day.yuv	268	0/0/0	0/0/0
4k	bund_night_scape.yuv	216	0/0/0	0/0/0
	campfire_party.yuv	216	0.3/2.3/0	0.3/2.3/0
	rush_hour.yuv	216	0/0/0	0/0/0
	tall_buildings.yuv	222	0/0/0	0/0/0
	tree_shade.yuv	225	0/0/0	0/0/0
	basketball.yuv	220	0/0/0	0/0/0
	construction.yuv	231	0/0/0	0/0/0
	flowers.yuv	227	0/0/0	0/0/0
	lake.yuv	219	0/0/0	0/0/0
	park.yuv	214	0/0/0	0/0/0

8. Conclusions

In this work we have presented a methodology to embed and retrieve a stego-message in high-resolution videos as FullHD and 4k, employing the latest video standard, namely: HEVC. In order to do this, there is no need to modify the standard, in comparison with other approaches in literature.

The encoding of several testing sequences with the HM-16.2 reference software and the x265 real-time encoder allows to conclude that the modification of the 2nd most significant bit in 4×4 luminance intra-blocks enables hiding information without losing quality in the video.

According to the obtained results, the implemented algorithm meets the expectations of security requirements, such as robustness against attacks, the ability to hide information and the low perceptibility of hidden data.

In the future, these techniques must be extended to allow the use of the inter-mode compression.

Declaration of Competing Interests

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgement

This paper has been supported by the EU (FEDER) and the Spanish MINECO under grant TIN 2015-65277-R and by Banco Santander under grant PR26/16-20B-1.

David Rodríguez is the main author of the paper and has implemented the framework to apply the embedding and retrieval methods. He has carried out the experiments to verify that the proposed techniques work properly and also he has written the draft version of the paper.

Alberto A. Del Barrio, Guillermo Botella and David Cuesta have advised on the design of the algorithms, selection of the related work and contributed to the paper organization and writing.

Supplementary materials

Supplementary material associated with this article can be found, in the online version, at doi:10.1016/j.compeleceng.2019.106541.

References

[1] Cisco. 2017. “Cisco visual networking index: forecast and methodology, 2016–2021”. Document ID:1465272001663118. Cisco VNI.
[2] Sullivan G, Ohm J, Han W-J, Wiegand T. Overview of the High Efficiency Video Coding (HEVC) standard. IEEE Trans Circuit Syst Video Technol Dec 2012;22(12):1649–68.
[3] Fraunhofer Heinrich Hertz Institute, 2017. “HEVC test model (HM)”. https://hevc.hhi.fraunhofer.de/HM-doc/. Accessed Nov. 05, 2016.
[4] x265 project. http://x265.org/. GNU GPL 2 license, source code available at: https://bitbucket.org/multicoreware/x265/wiki/Home, 2016.
[5] Idbeaa T, Abdul Samad S, Husain H. A secure and robust compressed domain video steganography for Intra- and inter-frames using Embedding-Based Byte Differencing (EBBD) scheme. PLoS ONE 2016;11(3).
[6] Nakajima K, Tanaka K, Matsuoka T, Nakajima Y. Rewritable data embedding on MPEG coded data domain. IEEE Int. Conf. Multimedia Expo 2005:682–5 2005.
[7] Wong K, Tanaka K, Takagi K, Nakajima Y. Complete video quality-preserving data hiding. IEEE Trans Circuit Syst Video Technol Oct. 2009;19(10):1499–512.
[8] Ma X, Li Z, Tu H, Zhang B. A data hiding algorithm for H.264/AVC video streams without intra-frame distortion drift. IEEE Trans Circuit Syst Video Technol 2010;20(10):1320–30.
[9] Wang P, Zheng Z, Ying J. A novel video watermark technique in motion vectors. Int Conf Audio Lang Image Process 2008:1555–9 2008.
[10] Wang R, Hu L, Xu D. A watermarking algorithm based on the CABAC entropy coding for H.264/AVC. J Comput Inf Syst 2011;7:2132–41.
[11] Xu D, Wang R, Wang J. Prediction mode modulated data-hiding algorithm for H.264/AVC. J Real-Time Image Process 2010;7(4):205–14.
[12] Yang G, Li J, He Y, Kang Z. An information hiding algorithm based on intra-prediction modes and matrix coding for H.264/AVC video stream. AEU - Int J Electron Commun 2011;65(4):331–7.
[13] Dutta T, Gupta HP. A robust watermarking framework for High Efficiency Video Coding (HEVC) - Encoded video with blind extraction process. J Vis Comun Image Represent 2016;38:29–44 C (July 2016).
[14] Swati S, Hayat K, Shahid Z. A watermarking scheme for High Efficiency Video Coding (HEVC). PLoS ONE 2014;9(8).
[15] Chang P, Chung K, Chen J, Lin C, Lin T. An error propagation free data hiding algorithm in HEVC intra-coded frames. 2013 Asia-Pacific Signal and Information Processing Association. Annual Summit and Conference; 2013.
[16] Wang J, Wang R, Li W, Xu D, Huang M. A large-capacity information hiding method for HEVC video. In: Proceedings of the 3rd International Conference on Computer Science and Service System; 2014.
[17] Jiang B, Yang G, Chen W. A cabac based HEVCc video steganography algorithm without bitrate increase. J Comput Inf Syst 2015;11:2121–30.
[18] Ferreira AM. An overview on hiding and detecting stego-data in video streams. System & Network Engineering. University of Amsterdam; 2015.
[19] Marçal ARS, Pereira PR. A steganographic method for digital images robust to RS steganalysis. Universidade do Porto; 2005.
[20] Fairchild M. Color appearance models. 3rd Edition. John Wiley & Sons; 2013.
[21] Xiph.org Foundation. 2017. Derf’s test media collection. https://media.xiph.org/video/derf/. Accessed Oct. 27, 2017.
[22] Elecard. 2019. Video compression Guru. https://www.elecard.com/videos. Accessed Jun. 29, 2019.
[23] Shanghai Jiao Tong University. 2018. SJTU Media Lab. http://medialab.sjtu.edu.cn/web4k/index.html. Accessed Jan. 31, 2018.
[24] Tampere University of Technology. 2018. Ultra Video Group. http://ultravideo.cs.tut.fi/#testsequences. Accessed Jul. 15, 2018.
[25] Yonsei University. 2019. Multimedia Computing and Machine Learning Group. http://mcml.yonsei.ac.kr/downloads/4kuhdvideoquality. Accessed Jun. 29, 2019.

D. Rodríguez Galiano received the B.*Sc*. degree in Computer Engineering from the Autonomous University of Madrid (Madrid, Spain), in 2009 and the M. *Sc*. degree in Computer Science Security from the International University of La Rioja (La Rioja, Spain), in 2016. He is PhD student at Complutense University of Madrid (Madrid, Spain). His-current research interests include steganography, watermarking and security on video.

A.A. Del Barrio received the Ph.D. degree in Computer Science from the Complutense University of Madrid (UCM), Madrid, Spain, in 2011. Since 2017, he has been an Interim Associate Professor of computer science with the Department of Computer Architecture and System Engineering, UCM. His-research interests include Design Automation, Arithmetic as well as Video Coding Optimizations. include applied probability and military operations research.

G. Botella received the Ph.D. degree in Computer Science from the University of Granada, Spain, in 2007. He was a research fellow funded by EU working at University College London, UK. Currently he is an Associate Professor at the Department of Computer Architecture and Automation of Complutense University, Madrid, Spain. His-research interests include Digital Signal Processing for FPGAs, GPUs and HPC.

D. Cuesta received the Ph.D. degree in Computer Science from the Complutense University of Madrid in 2013. Since 2010 he has been working for the railway sector, in charge of international IT projects, developing Passenger Information Systems and Signalling and Controlling Systems. He also received his Electronic engineering and Physics degree in 2008. His-current research are video coding and railways applications.