symmetry

Article

A Robust and High Capacity Data Hiding Method for H.265/HEVC Compressed Videos with Block Roughness Measure and Error Correcting Techniques

Kusan Biswas

School of Computer & Systems Sciences, Jawaharlal Nehru University, New Delhi 110067, India; kusan91_scs@jnu.ac.in

Received: 7 September 2019; Accepted: 18 October 2019; Published: 3 November 2019

Abstract: Recently, the H.265/HEVC video coding has been standardised by the ITU-T VCEG and the ISO/IEC MPEG. The improvements in H.265/HEVC video coding structure (CTU, motion compensation, inter- and intra-prediction, etc.) open up new possibilities to realise better data hiding algorithms in terms of capacity and robustness. In this paper, we propose a new data hiding method for HEVC videos. The proposed method embeds data in 4×4 and some selected larger transform units. As theory of Human Visual System suggests that human vision is less sensitive to change in uneven areas, relatively coarser blocks among the 8×8 and 16×16 blocks are selected as embedding destinations based on the proposed Jensen-Shannon Divergence and Second Moment (JSD-SM) block coarseness measure. In addition, the SME(1,3,7) embedding technique is able to embed three bits of message by modifying only one coefficient and therefore exhibits superior distortion performance. Furthermore, to achieve better robustness against re-compression attacks, BCH and Turbo error correcting codes have been used. Comparative studies of BCH and Turbo codes show the effectiveness of Turbo codes. Experimental results show that the proposed method achieves greater payload capacity and robustness than many existing state-of-the-art techniques without compromising on the visual quality.

Keywords: data hiding; H.265/HEVC; robustness; coding theory

1. Introduction

Digital motion picture, colloquially known as video, has become one of the most popular media in the global entertainment industry. It also has widespread use in IP telephony, surveillance and audio-visual education. The growing dependence on Internet and ubiquitous availability of network coverage through ADSL broadband, Wi-Fi, 3G and 4G LTE mobile Internet services together with cheap data packages has enabled almost exponential growth in usage of popular social media and content sharing services such as Facebook, YouTube, Instagram, Twitter, etc. The availability of high speed mobile Internet and the convenience of handheld mobile devices has given rise to online content sharing and streaming services such as Netflix, YouTube, Hulu, etc., which are rapidly replacing the traditional cable and satellite TV services. Due to the ease and convenience with which video contents can be downloaded and shared through social networking services over the Internet, the industry feels the need control, manage and protect their copyrights of their intellectual property. Specifically, the industry wishes: (1) to manage distribution rights of their video content to protect their business interest; (2) to monitor and keep track of when, where, how many times and by whom a copyrighted video has been downloaded, shared and streamed; and (3) to hyperlink the related contents together to enhance the end-user experience. Data hiding in video is one of the possible solutions that can fulfil the aforementioned objectives. Data hiding is the process of embedding private information inside public medium by altering some of its features. Apart from content protection and rights management, data hiding is also used in covert communication. The science of covert communication by means of hiding secret information in a public carrier medium is known as Steganography. In steganography, secret data are embedded in digital files in such a way that their presence is not perceptible. The carrier file is called the cover medium. The idea of steganography is to hide information inside cover medium such that its presence is not suspected in the first place, and even if it is suspected, can not be extracted by an unintended recipient or eavesdropper without the knowledge of a secret key. A successful steganography algorithm has to fulfil three main objectives: (1) the change in the cover medium due to secret data embedding is not perceptible; (2) the purpose and integrity of cover medium is maintained (e.g., a video cover file should be playable and not get corrupted due to data embedding, or else will raise suspicion); and (3) data should be recoverable by the intended recipient. In addition to these requirements, efforts are made to achieve high payload capacity, lower distortion, total recover-ability of embedded data, better resilience against transmission loss and security against statistical steganalysis attacks.

With growing popularity of online video, the demand for better video coding standard also arose. The industry needed more efficient video coding with better visual quality and better compression ratio. With the standardisation of H.264/AVC (technically known as ISO/IEC 14496-10 and ITU-T H.264) by ITU-T in 2003, the demands of the industry and the end users were partially met. Since then, with the increase in Internet penetration, Internet bandwidth and growing popularity of online streaming services (e.g., Youtube, Netflix, Hulu, etc.) and mobile video telephony, the industry has started to demand for even better video quality with better coding efficiency than the previous generation coding standards. Moreover, with large screen displays becoming ubiquitous, end users now demand beyond HD resolutions such as 4K (3840×2160 and 4096×2160) and 8K (7680×4320). To fulfil these demands, the MPEG and ITU-T VCEG have designed a new video coding standard called the H.265 High Efficiency Video Coding, commonly referred to as H.265/HEVC or MPEG-H Part 2. The first version of the H.265/HEVC standard was formally published in 2013. Since then, it has undergone several revisions and several extensions addressing specific applications have been added to it. The latest version, i.e., version 4, has been approved by the ITU-T in 2016. The H.265/HEVC standard has been designed to address four main issues: better compression ratio at the same perceptual quality compared to H.264/AVC, higher supported video resolution, better usage of parallel processing capabilities of modern CPUs/GPUs and less computationally intensive decoding. Since its standardisation, H.265/HEVC has been found to provide 50% better compression ratio than H.264/AVC at the same quality [1]. Due to this significant savings in bitrate, HEVC is already seeing widespread adoption in the industry and it is gradually replacing H.264/AVC in many application domains. In the coming years, H.265/HEVC is expected to be ubiquitous along with H.264/AVC if not totally replace the latter. Due to foreseen future popularity and widespread adoption of H.265/HEVC, and due to the importance of better data hiding methods for this new video coding technology, in this paper, we present a robust and secure data hiding method that achieves much larger payload capacity than earlier state-of-the-art methods, yet does not compromise on visual quality.

The rest of the paper is organised as follows. In Section 2, we discuss the related works and their weaknesses that motivate this research work. Section 3 gives a brief overview of H.265/HEVC and discusses its improvements and new features over the older video coding standards, especially the H.264/AVC. In Section 4, we discuss the theories of data encoding that have used in the proposed method. In Section 5, the proposed data hiding framework is presented. Next, in Section 6, the experimental results are presented and discussed. Finally, in Section 7, the paper is concluded with remarks on further research in this area.

2. Related Works and Motivation

In literature, many data hiding schemes (both steganographic and watermarking) have been proposed for H.264/AVC [2–10]. Recently, after the final standardisation of H.265/HEVC, many data hiding methods [2,11–13] have been proposed for this new standard. Similar to data hiding methods in image and early video compression standards such as H.264/AVC, data hiding methods for H.265/HEVC embed data by modifying one or more types compression features available in the H.265 coding standard. These include the quantised DCT and/or DST coefficients (i.e., the QTCs), motion vectors (MVs), coding block size pattern, intra-prediction modes, etc. Selection of appropriate type and number of embedding venue (e.g., QTCs) is done according to proposed models that aim to minimise embedding distortion and to increase payload. In addition to these, robustness against common attacks such as re-compression (i.e., re-quantisation) and image/video processing attacks (e.g., Gaussian filter, salt and pepper noise) are attempted to ensure by introducing redundancy in embedded data using error correcting codes such as Cyclic Redundancy Check (CRC), Hamming codes, BCH codes, Reed–Solomon codes, etc. In the following, we discuss some notable HEVC data hiding techniques proposed in the literature, whose merits and shortcomings motivate our work.

Chang et al. [2] discussed a method that embeds data only in the intra-coded frames of HEVC video sequence. It embeds data by introducing perturbation to the quantised DCT and DST coefficients of the luma prediction blocks. However, since only one feature is used (QTCs) in I-frames and since there is only one I-frame in each GOP of a video sequence, the overall payload capacity has room for improvement. Liu et. al. [12] discussed a data hiding method that avoids intra-frame distortion drift by selecting embedding destinations according to three conditions. These destinations are the DST coefficients in the luma channel of 4×4 blocks in I-frames. This method also suffers from low payload capacity as it embeds data only in the 4×4 blocks of only the I-frames. In [13], the authors discussed a scheme that embeds data in the non-zero quantised (NNZ) transform coefficients of the 4×4 intra-predicted blocks of the luminance channel. In this method, candidate 4×4 blocks for embedding are selected based on a number of criterion: number of NNZ coefficients, smoothness, motion information and quality. Next, a bipolar sequence of data is embedded in each candidate block by perturbing the values of the first two NNZ AC coefficients. The authors of [14] proposed a method that embeds data in 4×4 DCT blocks of some selected frames that met their proposed conditions to avoid distortion. Shamir's (t,n)-threshold secret sharing claimed to have improved robustness of the method, but the payload capacity is very low. In [15], the authors proposed a watermarking method for H.265/HEVC videos that embeds data in the LSBs of QTCs which are selected based on pre-defined thresholds. This method suffers from low payload capacity and robustness. The authors of [4] proposed a method that aims to minimise distortion by embedding data in paired-coefficients of the 4×4 blocks in selected frames. The authors also used BCH error correction coding to improve robustness. This method achieves good robustness but payload capacity is unsatisfactory. In [16], the authors proposed a method for HEVC videos that embeds data the 4×4 luma discrete sine transform blocks that meet their proposed grouping criteria for lower distortion. The authors also used BCH error correcting code to achieve robustness against re-quantisation attacks. However, both payload capacity and robustness have rooms for further improvement as only the 4×4 blocks are used as data embedding venues. Gaj et al. [17] proposed a watermarking technique that embeds data in the non-zero transform coefficients of the "motion-coherent" luma transform blocks that are selected based motion-vector information from neighbouring P-frames. This method yields good visual quality and robustness against salt and pepper, median-filter and Gaussian-filter attacks. However, payload capacity is not satisfactory as it embeds data in QTCs of only the of 4×4 blocks. Several techniques [18,19] have been proposed that embeds data in coding block size pattern. In [18], data are encoded in the video stream by adaptively manipulating the HEVC block size decision. The RDO (Rate Distortion Optimiser) module of HEVC estimates the cost function of distortion and bits required for all possible combinations of block sizes and takes decision what block size to allocate in a CTB structure in order for optimum compression ration with an specified video bit-rate. In [18], RDO is overridden and block size decision is manipulated to encode data bits in block-size pattern. Some additional data are embedded in the non-zero QTCs using odd–even LSB embedding technique. This method yields acceptable distortion and payload capacity, but the authors have not considered robustness in this work. Yang et al. [19] proposed a multilevel data hiding technique for HEVC videos that is based on PU partition modes in P-frames. This technique work in two passes. The first pass records the PU partition modes selected by the H.265 algorithm. The second pass is the modification pass. This pass forces each PU mode into one of the encoding groups to reflect the presence of the binary data bits. Finally, the modified PU partition modes are used in rest of the HEVC encoding process. This technique achieves good payload capacity at acceptable level of bit-rate increase. However, robustness of embedded data has not been evaluated by the authors. Apart from the above, several techniques [20–22] have been proposed in the literature that use the motion vector feature of HEVC as information hiding venue. In [20], the authors proposed a reversible data hiding method that embeds data in pairs of MVs by reducing difference expansion coefficient. The actual embedding is done by substituting the LSBs of the MVs. This technique yields good perceptual visual quality and acceptable level of bit-rate increase, but robustness performance was not considered in the paper. The authors of [22] proposed a data hiding technique that embeds in motion vector features of both P-frames and B-frames in compressed video. To minimise perceptual distortion, a candidate subset of MVs is selected based on prediction error. This approach yielded good distortion performance; however, the authors did not consider robustness performance.

The research work in this paper is motivated by observations about the existing works in data hiding in AVC and HEVC coded videos. Firstly, many data hiding methods have been developed for the H.264/AVC and earlier video compression standards. The literature of methods developed for H.264/AVC is very rich [3–10]. However, most of these existing methods cannot be applied to H.265/HEVC due to many changes in coding structure and prediction algorithms, which are briefly discussed in Section 3.1. Secondly, most of the aforementioned methods that use the QTCs as data embedding venue, only embed data in the 4×4 intra-predicted blocks and therefore suffer from low payload capacity. Thirdly, many proposed techniques that utilise MV and block partitioning as data embedding features do not consider robustness as a performance objective. The main objectives of this research work are to increase payload capacity by embedding data in larger prediction (8×8 and 16×16 in addition to the 4×4 blocks) unit in CTBs, to improve visual quality by minimising distortions caused by data embedding and to increase robustness by using powerful error correcting codes. The main contributions of the paper are as follows. Firstly, the Jensen-Shannon Divergence and Second Moment based block coarseness measure (JSD-SM) is proposed. JSD-SM coarseness measure selects the relatively more coarse blocks among all 8×8 and 16×16 luma TUs. Secondly, increased robustness is achieved using Turbo coding . Multiple sets of parameters of Turbo coding are proposed, evaluated and the results are compared to BCH codes. Thirdly, the Simplified Matrix Encoding, i.e., $\mathrm{SME}(1,3,7)$ data embedding technique, is able to embed three data bits in a block of seven quantised transform coefficients (QTCs) by modifying only one QTC. Hence, it reduces change density (i.e., number of modification per embedded data bit) of the proposed method.

3. Overview of H.265/HEVC Video Coding Standard

The H.265/HEVC video coding takes a hybrid approach of intra- and inter-frame prediction, motion compensation and 2-D discrete transform. The algorithm takes raw frames as input and divides the sequence into groups of pictures (GOP). Each GOP contains a pre-defined number of frames. The first frame of a GOP is coded using intra-picture prediction, i.e., using redundant spatial data within the same frame. The rest of the frames in a GOP are coded using inter-picture prediction in either of the two modes: (1) prediction mode, in which the ith frame is coded with taking advantage of temporal redundancy with the $(i-1)$th frame, resulting in P-frames; and (2) bi-directional prediction mode, which takes in to account both $(i-1)$th and $(i+1)$th frame while coding the ith frame and results in B-frames. Therefore, each GOP contains one I-frame in the beginning and the rest of the frames are either P-frames or B-frames. Inter-picture (P and B frames) encoding process involves finding motion information, i.e., motion vectors associated to each block of the current picture with respect to the reference frame, i.e., the previous I-frame. This process is known as motion compensation and is based on the idea that in a video sequence, many blocks in the picture are expected to move with respect to the I-frame, and therefore, to encode the next frames, it suffices to encode which blocks moved how much and in which direction, rather than encoding all the blocks themselves. The residual signal, i.e.,the difference between the original blocks and its predicted version, is mathematically transformed using a spatial transform. The transform coefficients are then passed through a series of processes involving scaling, quantisation and entropy coding. The final output is then stored or transmitted in a format specified by the HEVC standard. Figure 1 shows the overall encoding process.

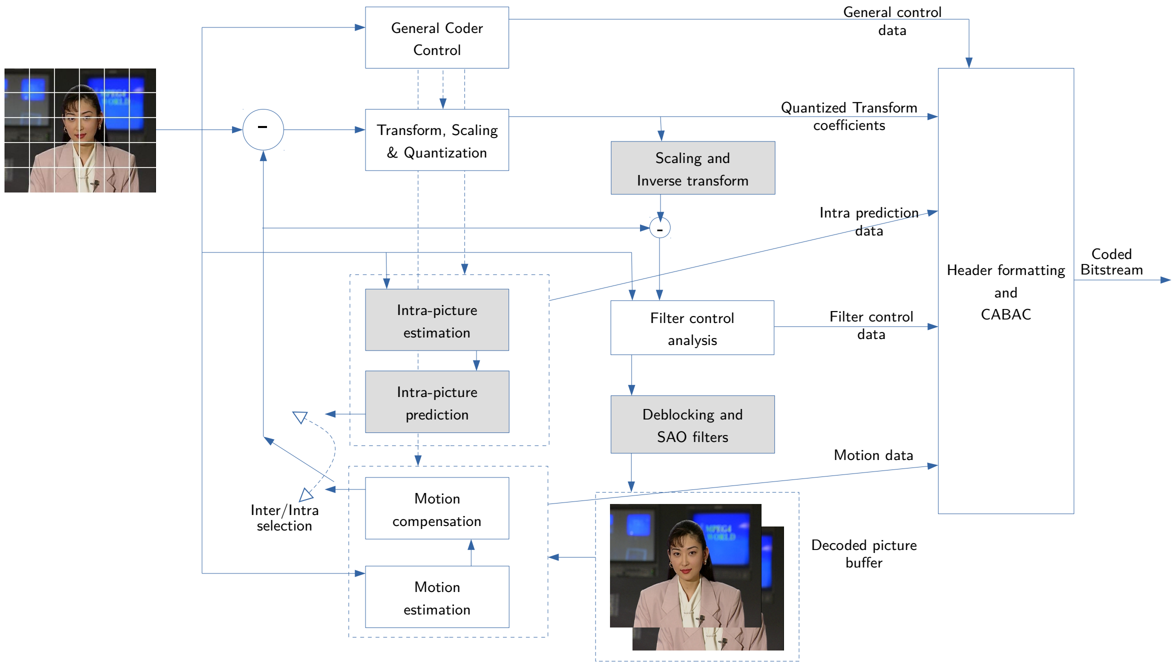

Figure 1. Overall HEVC encoding process.

3.1. *Improvements in H.265/HEVC over H.264/AVC*

3.1.1. Coding Unit

Each picture in RAW video sequence is partitioned into fixed sized blocks. In older standards (e.g., from H.261 to H.264/AVC), these blocks are called macroblocks (MBs). H.262 standard specified fixed 16×16 macroblocks. H.264/AVC macroblocks consist of 16×16 block of luminance (henceforth referred to as luma) samples and two corresponding blocks of chrominance (henceforth referred to as chroma) samples of size 8×8 in the typical 4:2:0 colour sampling arrangement. The sub-sampled 8×8 blocks contain samples of Cb and Cr components of the YCbCr colour space. Figure 2 summarises the macroblock sizes supported in H.264/AVC. MBs are the basic coding units in all video standards prior to H.265/HEVC. HEVC supports larger than 16×16 variable sized MBs and specifies a completely different tree-like organisation of MBs. The data structure in HEVC that is analogous to MBs in H.264, is the coding tree unit (CTU). The sizes of CTUs can be larger than the MBs of H.264. HEVC CTUs contain luma and chroma coding tree blocks (CTB) whose size can be as large as 64×64. HEVC supports these blocks to be further divided and arranged in quad-tree structure, as shown in Figure 3.

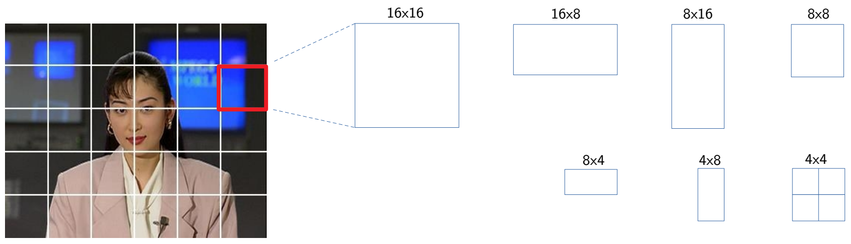

Figure 2. H.264/AVC macroblocks and their segmentation into smaller blocks.

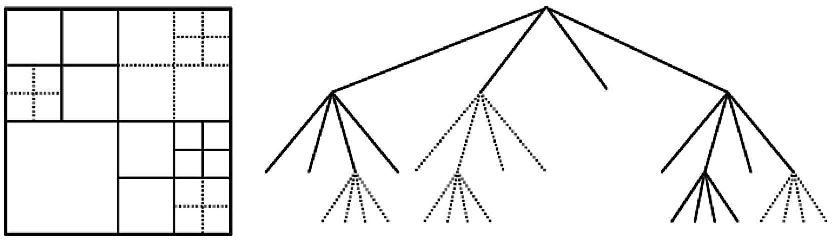

Figure 3. (**Left**) HEVC coding tree unit (CTU) and its division into smaller coding blocks (CB) and transform blocks (TB); and (**Right**) the CTU quad-tree.

3.1.2. Motion Estimation

Motion estimation is the process of finding matching blocks of pixels in inter-frame coding. In intra-frame coding, although there is no motion involved as the blocks are matched within a single frame, the objective is same: to take advantage of data redundancy. Intra prediction finds spatial redundancy, whereas inter-frame prediction estimates motion of pixel blocks to find temporal redundancy between two consecutive frames. In intra-prediction, i.e., when utilising spatial redundancy for compression within a I-frame, the H.264/AVC supports nine prediction modes. These modes are shown in Figure 4. For large 16 × 16 luma blocks, four4 prediction modes are supported. H.265/HEVC significantly improved intra-prediction by introducing much finer angles of supported directions. HEVC specifies 33 non-uniform angular prediction modes (Figure 5). The angles are finer at near-horizontal and near-vertical angles and more coarse at diagonal angles. This arrangement provides better statistical matches of blocks of pixels across frames.

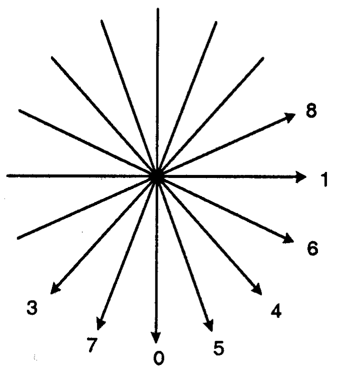

Figure 4. The nine intra-prediction modes for 4 × 4 luma blocks in H.264/AVC.

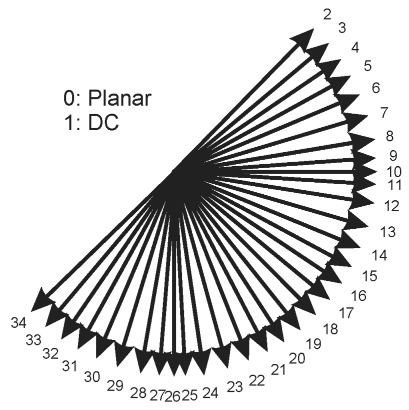

Figure 5. The 33 angular intra-prediction modes in H.265/HEVC.

3.1.3. Transform Coding and Quantisation

Two types of transforms have been specified in the H.265/HEVC standard [23]: core transform and an alternative transform [24,25]. The core transform is the discrete cosine transform (DCT) that is performed on $4 \times 4, 8 \times 8, 16 \times 16$ and 32×32 transform blocks (TB). However, H.265 standard specifies only one transform matrix of size 32×32 [23]. The rest of the transform matrices for smaller TBs ($4 \times 4, 8 \times 8$ and 16×16) are computed by sub-sampling the 32×32 transform matrix [24,26]. The alternative transform, which is used only for 4×4 luma residual TBs in the intra-picture prediction mode, is derived from discrete sine transform (DST). The 4×4 DST is computationally not too much more demanding than DCT of same dimension. However, on average, DST results in approximately 1% savings in terms of bit rate. The sub-sampled 8×8 core transform matrix and the 4×4 alternative transform matrices are shown in Equations (1) and (2)

$$D_{inter\_8\times8\_DCT} = \begin{bmatrix} 64 & 64 & 64 & 64 & 64 & 64 & 64 & 64 \\ 89 & 50 & -18 & -75 & -89 & -50 & 18 & 75 \\ 83 & -36 & -83 & 36 & 83 & -36 & -83 & 36 \\ 75 & -89 & 50 & 18 & -75 & 89 & -50 & -18 \\ 64 & -64 & 64 & -64 & 64 & -64 & 64 & -64 \\ 50 & 18 & -75 & 89 & -50 & -18 & 75 & -89 \\ 36 & 83 & -36 & -83 & 36 & 83 & -36 & -83 \\ 18 & 75 & 89 & 50 & -18 & -75 & -89 & -50 \end{bmatrix} \tag{1}$$

$$D_{intra\_4\times4\_DST} = \begin{bmatrix} 29 & 55 & 74 & 84 \\ 74 & 74 & 0 & -74 \\ 84 & -29 & -74 & 55 \\ 55 & -84 & 74 & -29 \end{bmatrix} \tag{2}$$

3.1.4. Entropy Coding

The MPEG-1 and MPEG-2 standards employed Huffman coding as their entropy coding model. H.264/AVC specifies two entropy coding models: context-adaptive binary arithmetic coding (CABAC) and context-adaptive variable length coding (CAVLC) for its baseline and main profiles respectively [27]. H.265/HEVC specifies only CABAC for all profiles [23]. CABAC is a lossless entropy coding model that employs different probability models for different contexts. This allows better modelling of distribution as local data are generally well-correlated. Although CABAC is more complex and computationally intensive than CAVLC and Huffman coding, it is 10% more efficient than CAVLC in terms of bitrate savings [28].

4. Overview of Error Correcting Techniques

Several researchers [14,29–32] have shown that error correcting codes (linear block and convolutional codes) such as CRC, Reed–Muller codes, and Bose–Chaudhury–Hocquenghem (BCH) codes can significantly reduce embedding distortion and increase robustness in noisy and unreliable transmission scenario and also against video re-quantisation attacks. In this paper, we use a carefully designed set of BCH codes and Turbo codes [33]. In the following sections, we briefly discuss the theories of BCH and Turbo codes.

4.1. BCH Syndrome Error Correcting Codes

Bose–Chaudhury–Hocquenghem (BCH) is a powerful class of random error correcting codes that is capable of correcting multiple errors. For any integers $m \geq 3$ and $t \leq (2^m - 1)/2$, BCH codes are characterized by the following parameters:

$$n = 2^m - 1 \tag{3}$$

$$k \geq n - mt \tag{4}$$

$$d_{min} \geq 2t + 1 \tag{5}$$

where n is the code block length, $(n - k)$ is the number of parity check bits and d_{min} is the minimum distance. This system of code is capable of correcting t-errors and referred to as $BCH(n, k, t)$. If α is a primitive element of the Galois Field $GF(2^m)$, m being the order of the field, n being the length of the codeword, and k being the dimension of the code, the error correction matrix H for $BCH(n, k, t)$ is as follows:

$$H = \begin{bmatrix} 1 & \alpha & \alpha^2 & \dots & \alpha^{n-1} \\ 1 & \alpha^3 & (\alpha^3)^2 & \dots & (\alpha^3)^{n-1} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & \alpha^{2t-1} & (\alpha^{2t-1})^2 & \dots & (\alpha^{2t-1})^{n-1} \end{bmatrix} \tag{6}$$

Given the original data are $D = \{d_0, d_1, d_2, \dots, d_{k-1}\}$, the BCH codeword $V = \{v_0, v_1, v_2, \dots, v_{n-1}\}$ for D is calculated as follows:

$$V = DH \tag{7}$$

An unreliable, noisy channel may introduce multiple errors in the data. If $V' = \{v'_0, v'_1, v'_2, \dots, v'_{n-1}\}$ is the erroneous data received at the receiver end and if E is the error pattern, then

$$V = V' + E \tag{8}$$

The syndrome S is given by the $2t$-tuple:

$$S = \{s_0, s_1, s_2, \dots, s_{2t-1}\} \tag{9}$$

Let $s_i = v'(\alpha_i) = v'_0 + v'_1\alpha^i + \dots + v'_{n-1}(\alpha^i)^{n-1}$ for $1 \leq i \leq 2t$. If $\phi(x)$ is the minimal polynomial of α^i, for binary polynomial, it holds that,

$$r(x) = a_i(x)\phi_i(x) + b_i(x) \tag{10}$$

Hence, s_i can be evaluated as $s_i = b_i(\alpha_i)$. Since $v'(\alpha^i) = 0$ for $1 \leq i \leq 2t$, $s_i = r(\alpha^i) = e(\alpha^i)$ where $e(x) = x^{j_1} + x^{j_2} + \ldots + x^{j_k}$ is the error pattern, such that $0 \leq j_1 \leq j_2 \leq \ldots \leq n-1$. Therefore,

$$\begin{aligned} s_1 &= \alpha^{j_1} + \alpha^{j_2} + \ldots + \alpha^{j_k} \\ s_2 &= (\alpha^{j_1})^2 + (\alpha^{j_2})^2 + \ldots + (\alpha^{j_k})^2 \\ &\vdots \\ s_{2t} &= (\alpha^{j_1})^{2t} + (\alpha^{j_2})^{2t} + \ldots + (\alpha^{j_k})^{2t} \end{aligned} \tag{11}$$

where $\alpha^{j_1}, \alpha^{j_2}, \ldots, \alpha^{j_k}$ are unknown. To decode BCH code, we must solve Equation (11). In this paper, we use Berelkamp's iterative algorithm [34] as BCH decoding algorithm.

4.2. Turbo Codes

To increase coding efficiency of the traditional codes, and to approach the Shannon limit [35], the code-word length of the linear block codes (or constraint length, in the case of convolutional code) should be increased. However, increasing the code-word length causes the complexity of the decoder to increase exponentially and the decoder takes proportionately longer time. Turbo codes were proposed to address these issues. Turbo code [35] is an approach that simulates larger coding blocks by splitting and interleaving it, such that decoding can be done in a number of manageable steps. An interleaver is a component that temporally permutes (with the help of a memory buffer) a sequence of symbols in a totally deterministic manner. An added benefit of interleaving is that burst errors in data can be converted to statistically independent short errors when the data are de-interleaved at the decoder side. This enables code designed for statistically independent errors to be used as the constituent codes (for encoder 1 and 2). Figure 6 depicts the basic building blocks of a Turbo encoder. The interleaver permutes (in periodic or pseudo-random manner) the input bits such that the two encoders operate on the same set of input bits, but in different order. The constituent encoders may use different or same convolution or linear block codes. Figure 7 shows the general structure of a Turbo decoder.

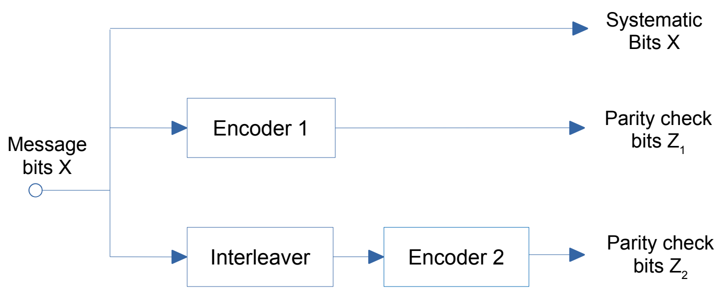

Figure 6. Basic building blocks of a Turbo encoder.

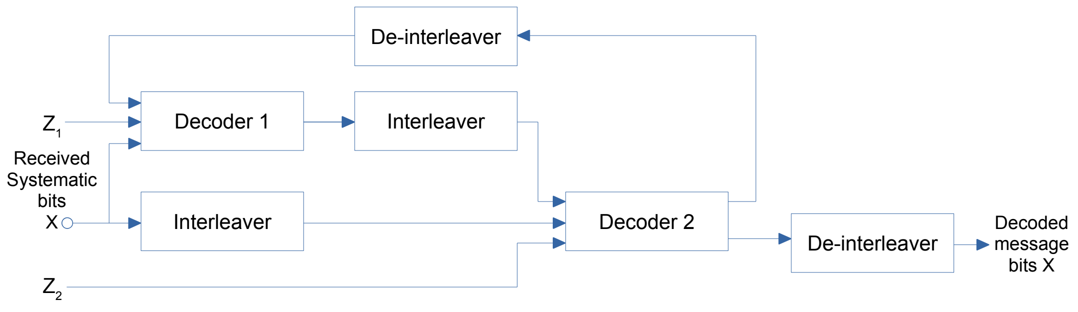

Figure 7. Block diagram of a general Turbo decoder.

5. Proposed Method of Data Hiding

As discussed in Section 2, the proposed method focuses on improving three main aspects of data hiding methods: payload capacity, robustness against transmission errors or re-quantisation attacks and distortion performance. As discussed in Section 2, most of the existing works in the literature embed data in only the 4×4 blocks as these are the most irregular blocks. Human Visual System is not very sensitive to little changes in rough areas, therefore embedding in 4×4 blocks does not cause noticeable visual degradation. However, this severely limits payload capacity. To increase payload capacity, in this paper, we embed data in selected 8×8 and 16×16 TU blocks in addition to all 4×4 blocks. To keep visual distortion under control, we embed data only in the most irregular 8×8 and 16×16 blocks. These blocks are selected using the proposed JSD-SM block coarseness measure. In addition, most existing methods in the literature [2–13] embed data by directly modifying the LSBs of the QTC values. In these methods, the LSB of one QTC is modified to embed one bit of data. The SME(1,3,7) technique presented in this paper, is able to embed three bits of data by modifying one QTC in a block of seven QTCs. Therefore, in a 4×4 block, SME(1,3,7) is able to embed 6 data-bits by modifying only 2 QTCs. The change density is the density of data embedding, which is defined as the number of modifications per embedded data bit. In most existing literature [2–13], change density is 1 modification per 1 embedded data bit. The SME(1,3,7) technique is able to embed 3 data bits by modifying only 1 coefficient. Therefore, change density of SME(1,3,7) is 1/3, which is one-third of the existing methods. Reduced change density allows us to embed data in larger than 4×4 blocks without causing any significant visual distortion. However, we only select the relatively rough 8×8 and 16×16 blocks based on the proposed JSD-SM coarseness measure. The video stream with embedded data would most likely be transmitted over the network or streamed in publicly shared channels. Most transmission channels (e.g., WiFi and mobile broadband networks) are prone to error. This error may cause some coefficients of the compressed video to change. Moreover, an attacker may deliberately introduce perturbation in the QTCs or recompress (i.e., re-quantise) the video to foil correct recovery of embedded secret data by the intended receiver. To attain a high degree of robustness against these attacks, in this paper, we use BCH and Turbo error correcting codes. The BCH code has been used in some existing literature [3,4,30,31]. However, its effectiveness is expected to increase when used in conjunction with the SME(1,3,7) whose change density is one-third of most of the existing LSB based embedding techniques. In addition to BCH codes, we also use Turbo error correcting codes. The BCH codes are able to correct random single bit errors but are less effective against burst errors, i.e., when more than one consecutive bits are changed due to error. Turbo codes are a family of powerful errors correcting codes that have been developed to correct such burst errors in addition to random single bit errors. In the following subsections, we present the proposed JSD-SM block coarseness measure, the SME(1,3,7) data embedding and extraction technique and design, and parameters of the BCH and Turbo error correcting codes. In Section 5.4, we present the overall embedding and extraction process.

5.1. Block Selection Using JSD-SM Coarseness Measure

As discussed in Section 3.1.1, the H.265/HEVC compression algorithm partitions the frames in transform units (TU) of size from 4×4 to 32×32 which are organised in CTB structures. HEVC partitions the picture frame in different TU sizes based on smoothness or coarseness of a region. If a region in a picture is coarse, i.e., contains high variation of pixel values, it is subdivided in many 4×4 blocks to retain maximum detail. Larger blocks such as 8×8, 16×16 and 32×32 are allocated for relatively less coarse regions. In the proposed work, message bits are embedded in all 4×4 blocks since these are the most coarse blocks. However, to achieve higher payload capacity, it is necessary to embed data not only in $4\times$, but in larger 8×8 and 16×16 blocks too. As the HEVC algorithm allocates larger blocks to the relatively less coarse regions, the 8×8 and 16×16 blocks are relatively less coarse than the 4×4 blocks. Embedding data in these blocks may introduce visual artefacts as the human visual system is sensitive to small changes in smooth (i.e., low frequency) regions. To minimise embedding distortions in these blocks, we embed data in the relatively most coarse blocks among all 8×8 and 16×16 blocks.

To select the coarser blocks, we propose a method that is based on Jensen–Shannon divergence (JS-divergence) [36] measure and second moment of the pixel values. We call the proposed method block selection using Jensen–Shannon Divergence and Second Moment (JSD-SM). Jensen–Shannon divergence is a symmetric adaptation of Kullback–Leibler divergence (KL-divergence). Given two probability distributions P and Q in the same probability space, the KL-divergence is defined as,

$$D_{KL}(P\|Q) = \sum_{x \in X} P(x) \log \frac{P(x)}{Q(x)} \tag{12}$$

Jensen–Shannon divergence is then defined as,

$$D_{JS}(P\|Q) = \frac{1}{2} D_{KL}(P\|M) + \frac{1}{2} D_{KL}(Q\|M) \tag{13}$$

where $M = \frac{1}{2}(P + Q)$. Unlike KL-divergence, Jensen–Shannon divergence is finite, i.e., $0 \leq D_{JS} \leq 1$, and symmetric, i.e., $D_{JS}(P\|Q) = D_{JS}(Q\|P)$. These properties make it more suitable for our purpose.

The first and second moments of a block are defined as follows. Given a block B of dimension $m \times n$, its set of pixel values can be imagined as a one dimensional vector $P = \{x_1, x_2, x_3, \ldots, x_N\}$, where $N = mn$. The probability of a pixel value x_i appearing in B is $p_i = k/N$, where k is the number of times the x_i occurs in B. It is clear that $0 \leq p_i \leq 1$ and $\sum p_i = 1$. Hence, this is a valid probability distribution. The mean or the expectation of this distribution is $\mu = E[x] = \sum_{i=1}^{N} x_i p_i = x_1 p_1 + x_2 p_2 + \ldots + x_N p_N$. The central second moment or the variance is,

$$Var(X) = \sum_{i=1}^{N} p_i \, (x_i - \mu)^2 \tag{14}$$

Given the above definitions of Jensen–Shannon Divergence and second moment of probability distribution, we propose the JSD-SM block coarseness measure as follows (Algorithm 1).

Algorithm 1: JSD-SM block coarseness algorithm

```
Data: An $8 \times 8$ or $16 \times 16$ TU block denoted as $B$
Result: $C_{JSD-SM}$, the JSD-SM coarseness measure of the block
1  Initialise $SDB$ = spatial domain block corresponding to $B$;
2  Initialise number of pixels $N = |SDB|$;
3  Divide $SDB$ in $n$ number of $4 \times 4$ blocks $b_i, i = 1, \ldots, n$;
4  while there is a $4 \times 4$ block remaining do
5      read next block;
6      measure pixel value variance of $b_i$, $var_i = Var(b_i)$;
7  end
8  Let $avgVar_B = mean(var_i)$;
9  Let $p_i = k/N$, where $k$ is the number of times a pixel value occurs in $SDB$;
10 Clearly, $0 \leq p_i \leq 1$ and $\sum p_i = 1$. Denote this distribution as $P$;
11 Let $U$ be a uniform distribution with equal number of observations as $P$;
12 Find $D_{JS}(P||U)$ using (13);
13 Output $C_{JSD-SM} = avgVar_B \times (1 - D_{JS})$
```

5.2. Data Embedding and Extraction

The process of representing message bits by altering the carrier file features is called data embedding and the process of recovering embedded data from the cover medium is called data extraction. In the literature, many data embedding techniques have been proposed for steganography, watermarking and data hiding algorithms in general. The earliest and simplest data embedding technique is the Least Significant Bit (LSB) substitution [37] method and its derivatives. In LSB substitution based methods, one or more LSBs of embedding feature (e.g., pixel value, transform coefficient, motion vector, etc.) are substituted with the secret data bits. Equation (15) mathematically represents the basic LSB substitution based data embedding operation.

$$Y_i = 2\left\lfloor \frac{X_i}{2} \right\rfloor + m_i \tag{15}$$

where m_i is the ith message bit, X_i is the value of the ith selected pixel before embedding and Y_i is the value of the ith pixel after embedding [38]. LSB based techniques are simple, fast and generally have very good payload capacity. However, these techniques suffer from high distortion and lack of robustness against re-compression attacks and also suffers in presence of transmission errors. Several derivatives of LSB technique, namely LPAP [37], OPAP [39], SLSB [40] and PVD [41], improved the distortion performance, however these methods are not suitable for data embedding in video features due to following reasons: (1) low embedding efficiency [42]; (2) high distortion; (3) lack of robustness in erroneous transmission scenario; and (4) not robust against re-quantisation attacks. Lack of robustness of simple LSB substitution based techniques motivates us to use error correcting codes.

In this paper, for data embedding in the selected quantised transform coefficients (QTCs), we propose a Simplified Matrix Embedding (SME) technique. Matrix embedding techniques were discussed by Fridrich and Soukul [43]. SME technique embeds k data bits in a block of $n = 2^k - 1$ QTCs by modifying at most 1 QTC value. Specifically, in our case, we use a SME$(1, k = 3, n = 7)$ scheme. This scheme embeds three message bits in a block of seven QTCs. The embedding scheme works as follows.

1. Let a 3-bit message block be $M = (m_1 m_2 m_3)$ and the destination block of seven QTCs is $QB = (Q_1, Q_2, Q_3, Q_3, Q_4, Q_5, Q_6, Q_7)$. Only one of the Q_is is modified to encode the message block in QTC block.
2. Define three parity values P_1, P_2 and P_3 as follows.

$$P_1 = (Q_1 + Q_3 + Q_5 + Q_7) \ mod\ 2 \tag{16}$$

$$P_2 = (Q_2 + Q_3 + Q_6 + Q_7) \ mod\ 2 \tag{17}$$

$$P_3 = (Q_4 + Q_5 + Q_6 + Q_7) \ mod\ 2 \tag{18}$$

3. To encode binary message bits $(m_1 m_2 m_3)$, modify the QTC values according to the following rules:

Case 1. If $(m_1 = P_1) \wedge (m_2 = P_2) \wedge (m_3 = P_3)$, modify no QTC

Case 2. If $(m_1 \neq P_1) \wedge (m_2 = P_2) \wedge (m_3 = P_3)$, modify Q_1 as follows. If $|Q_1| = Q_{max}$, $|Q_1| = |Q_1 - 1|$. Else, $Q_1 = Q_1 + 1$

Case 3. If $(m_1 = P_1) \wedge (m_2 \neq P_2) \wedge (m_3 = P_3)$, modify Q_2 as follows. If $|Q_2| = Q_{max}$, $|Q_2| = |Q_2 - 1|$. Else, $Q_2 = Q_2 + 1$

Case 4. If $(m_1 \neq P_1) \wedge (m_2 \neq P_2) \wedge (m_3 = P_3)$, modify Q_3 as follows. If $|Q_3| = Q_{max}$, $|Q_3| = |Q_3 - 1|$. Else, $Q_3 = Q_3 + 1$

Case 5. If $(m_1 = P_1) \wedge (m_2 = P_2) \wedge (m_3 \neq P_3)$, modify Q_4 as follows. If $|Q_4| = Q_{max}$, $|Q_4| = |Q_4 - 1|$. Else, $Q_4 = Q_4 + 1$

Case 6. If $(m_1 \neq P_1) \wedge (m_2 = P_2) \wedge (m_3 \neq P_3)$, modify Q_5 as follows. If $|Q_5| = Q_{max}$, $|Q_5| = |Q_5 - 1|$. Else, $Q_5 = Q_5 + 1$

Case 7. If $(m_1 = P_1) \wedge (m_2 \neq P_2) \wedge (m_3 \neq P_3)$, modify Q_6 as follows. If $|Q_6| = Q_{max}$, $|Q_6| = |Q_6 - 1|$. Else, $Q_6 = Q_6 + 1$

Case 8. If $(m_1 \neq P_1) \wedge (m_2 \neq P_2) \wedge (m_3 \neq P_3)$, modify Q_7 as follows. If $|Q_7| = Q_{max}$, $|Q_7| = |Q_7 - 1|$. Else, $Q_7 = Q_7 + 1$

At the receiver end, data are extracted from the cover medium as follows.

1. Let a modified block of QTCs be $QB' = (Q'_1, Q'_2, Q'_3, Q'_3, Q'_4, Q'_5, Q'_6, Q'_7)$.
2. The parity conditions of the modified QTCs of the received cover medium are calculated with Equations (16)–(18) as follows:

$$P'_1 = (Q'_1 + Q'_3 + Q'_5 + Q'_7)\ mod\ 2 \tag{19}$$

$$P'_2 = (Q'_2 + Q'_3 + Q'_6 + Q'_7)\ mod\ 2 \tag{20}$$

$$P'_3 = (Q'_4 + Q'_5 + Q'_6 + Q'_7)\ mod\ 2 \tag{21}$$

3. Three message bits (m'_1, m'_2, m'_3) are extracted from QB' as follows.

$$m'_1 = P'_1,\ m'_2 = P'_2,\ m'_3 = P'_3$$

Illustration

Let the first seven coefficients in a 4×4 block of DST QTCs be $QB_1 = (Q_1, Q_2, Q_3, Q_3, Q_4, Q_5, Q_6, Q_7) = (5, 21, 12, 9, 0, 7, 18)$. The three bits message to be embedded are $m_1m_2m_3 = 011$. The parities are calculated with Equations (16)–(18) as,

$$P_1 = (5 + 12 + 0 + 18)\ mod\ 2 = 1$$

$$P_2 = (21 + 12 + 7 + 18)\ mod\ 2 = 0$$

$$P_3 = (9 + 0 + 7 + 18)\ mod\ 2 = 0$$

In this case, $m_1 \neq P_1, m_2 \neq P_2$ and $m_3 \neq P_3$. Hence, Case 8 is satisfied. Therefore, Q_7 is modified to $Q'_7 = (Q_7 + 1)$. Therefore, the modified block of QTCs becomes $QB'_1 = (Q_1, Q_2, Q_3, Q_3, Q_4, Q_5, Q_6, Q_7) = (5, 21, 12, 9, 0, 7, 19)$. Note that the three-bit message $m_1m_2m_3 = 011$ has been embedded by modifying only one QTC value.

At the receiver end, $QB'_1 = (5, 21, 12, 9, 0, 7, 19)$ is received and the modified parities P'_1, P'_2 and P'_3 are evaluated using Equations (19)–(21).

$$P'_1 = (5 + 12 + 0 + 19)\ mod\ 2 = 0$$

$$P'_2 = (21 + 12 + 7 + 19)\ mod\ 2 = 1$$

$$P'_3 = (9 + 0 + 7 + 19)\ mod\ 2 = 1$$

Thus, the recovered three bit message is 011, which matches the originally embedded message bits.

The next block of seven QTCs is $QB_2 = (Q_1, Q_2, Q_3, Q_3, Q_4, Q_5, Q_6, Q_7) = (10, 88, 12, 13, 9, 1, 6)$. Here, parities are $P_1 = 1$, $P_2 = 1$ and $P_3 = 1$. The message bits to be embedded are $m_1m_2m_3 = 100$.

In this case, $m_1 = P_1, m_2 \neq P_2$ and $m_3 \neq P_3$; therefore, Case 7 is satisfied. Hence, Q_6 is modified. The modified QTC block is $QB_2' = (10, 88, 12, 13, 9, 2, 6)$. At the receiver end, the modified parities are evaluated to find $P_1' = 1$, $P_2' = 0$ and $P_3' = 0$. Therefore, the received three-bit message is $m_1'm_2'm_3' = P_1'P_2'P_3' = 100$, which matches the originally embedded message. These two cases of embedding using SME(1,3,7) technique are illustrated in Figure 8 by taking a 4 × 4 HEVC/H.265 TU.

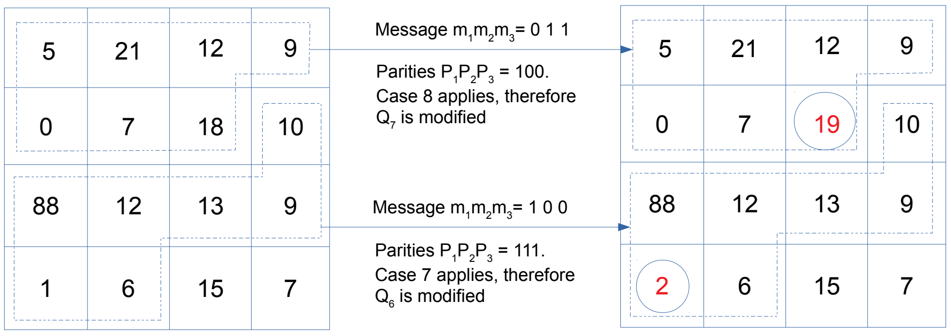
5 21 12 9
0 7 18 10
88 12 13 9
1 6 15 7
Message $m_1m_2m_3$= 0 1 1
Parities $P_1P_2P_3$ = 100.
Case 8 applies, therefore Q_7 is modified
Message $m_1m_2m_3$= 1 0 0
Parities $P_1P_2P_3$ = 111.
Case 7 applies, therefore Q_6 is modified
5 21 12 9
0 7 19 10
88 12 13 9
2 6 15 7

Figure 8. Embedding two groups of three bits of messages in blocks of seven QTCs by modifying only two QTCs using the SME(1,3,7) technique. The original 4 × 4 block is on the left. On the right, the modified QTCs are shown in circles.

5.3. Design of the BCH and Turbo Error Correcting Codes

We use two different data encoding algorithms for error correction codes: BCH and Turbo codes. These error correcting codes introduce redundancies that help correct errors in video stream that is transmitted through an error-prone channel or the errors that may be introduced due to re-quantisation attacks. This increases robustness or survivability of the embedded data. As discussed in Section 4.1, BCH coding schemes are described as BCH(n, k, t) where n is the code block length, k is the number of message bits and t is the number of error bits that the code is capable of correcting. We propose the following three set of parameters and corresponding generating polynomials of BCH(n, k, t) coding given in Table 1. In this paper, we use Berelkamp iterative algorithm [44] to decode BCH coded data.

Table 1. BCH coding schemes with three different sets of values of parameters n, k and t used in the proposed work.

Name	n	k	t	Generator Polynomial
BCH$(7,4,1)$	7	4	1	1 011
BCH$(31,16,3)$	31	16	3	1 000 111 110 101 111
BCH$(31,11,5)$	31	11	5	101 100 010 011 011 010 101

For Turbo coding, we employ two identical parallel concatenated convolution encoders (PCCE) as the constituent encoders in conjunction with a pseudo-random intervealer. The proposed structure of the constituent encoder is as shown in Figure 9. The constraint length of the Turbo code is determined by the pseudo-random interleaver. The decoder is the Soft-Output Viterbi algorithm (SOVA) [45] that decodes received codes by estimating logarithm of likelihood ratio (LLR) as in Equation (22).

$$\Lambda = log \frac{p(X = 1|R)}{p(X = 0|R)} \tag{22}$$

where R is the received bit. For constituent convolutional encoders, we propose a recursive convolutional code (RSC) that is used for both for CE_1 and CE_2. The structure of the constituent RSC is as shown in Figure 9. The proposed RSC has $2^3 = 8$ states and has constraint length of 4. The transfer function is as follows.

$$G(D) = \left[1, \frac{g_1(D)}{g_0(D)}\right] \tag{23}$$

where $g_0(D) = 1 + D^2 + D^3$ and $g_1(D) = 1 + D + D^3$. The initial values of the three shift registers are kept all zeros when starting the encoding process. The constraint length of Turbo encoders depend on the interleaver used. In this paper, we use a pseudo-random interleaver. The proposed interleaver is a linear congruential generator (LCG) given by Equation (24).

$$X_{n+1} = (aX_n + c) \ mod \ m \tag{24}$$

where X is the output sequence of pseudo-random numbers, $m\ (> 0)$ is the modulus, $a \in (0, m)$ is a constant multiplier, $c \in [0, m)$ is the increment and $X_0 \in [0, m)$ is the seed or the start value. The set of of these values and the corresponding constraint lengths K_t that are used in this paper are summarised in Table 2. The block diagram of the proposed Turbo encoder and decoder is illustrated in Figure 10 in which the convolutional encoders CE_1 and CE_2 are having the same structure as shown in Figure 9.

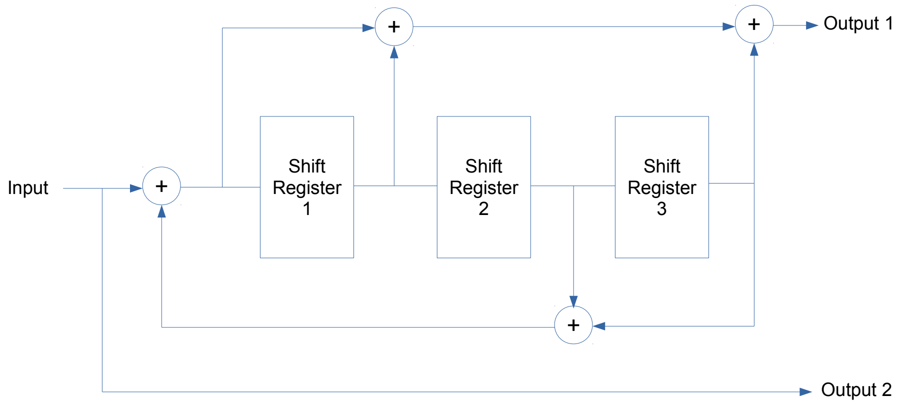

Figure 9. Block diagram of the half-rate, 8-state constituent recursive convolution encoder CE_1 and CE_2 with constraint length 4.

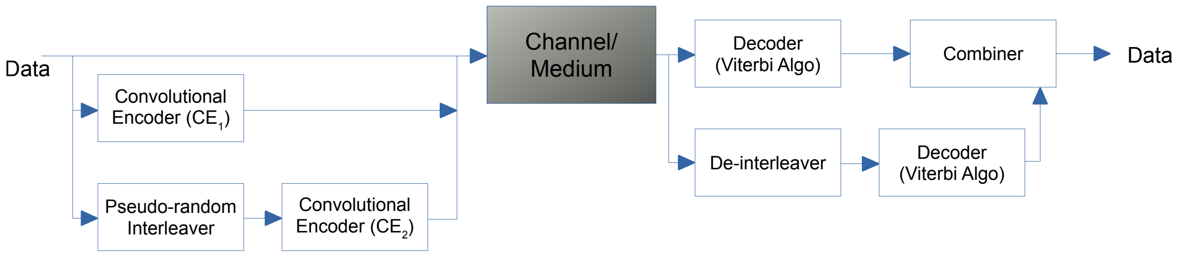

Figure 10. Block diagram of the proposed Turbo coding/decoding system.

Table 2. Turbo coding schemes with two different sets of values of parameters n, k and t and constraint length used in the proposed work.

Name	**Seed** (X_0)	m	a	c	**Sequence**	**Constraint Length** K_t
Turbo-16	3	5	3	2	$\{3, 1, 0, 2\}$	16
Turbo-24	2	7	3	0	$\{2, 6, 4, 5, 1, 3\}$	24

5.4. Overall Architecture of the Proposed Method

The proposed method embeds message bits into the H.265/HEVC quantised transform coefficients of the intra-coded frames. The embedding process is described in Algorithm 2 and illustrated in Figure 11. Once data are embedded in a video sequence, it is transmitted to one or more receiver over a public channel. At the receiver end, the receiver extracts the embedded data using the key. The receiver starts decoding the H.265/HEVC compressed video. After entropy decoding, all 4×4, 8×8 and 16×16 TU blocks in the luminance channel are copied and stored separately. Then, the usual HEVC inverse quantisation and inverse transform processes continues. Inverse transform generates the spatial domain blocks corresponding to each DST (4 × 4) and DCT (8 × 8 and 16 × 16) domain blocks. The JSD-SM coarseness of these spatial domain blocks are evaluated. Next, κ % most coarse blocks are selected. Data are the extracted from the quantised DST and DCT coefficients the corresponding transform domain blocks (that were save separately) using the SME(1,3,7) extraction process described in Section 5.2. After extraction, data is decoded and output to the receiver. The whole process is described in Algorithm 3 and illustrated in Figure 12.

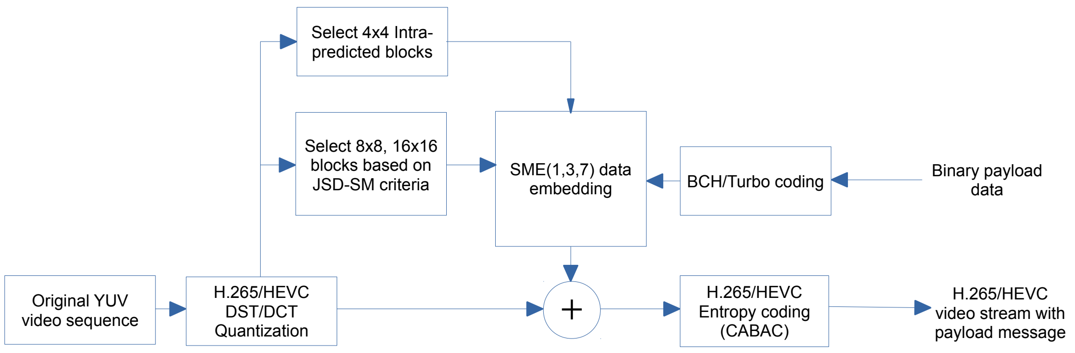
Select 4x4 Intra-predicted blocks
Select 8x8, 16x16 blocks based on JSD-SM criteria
SME(1,3,7) data embedding
BCH/Turbo coding
Binary payload data
Original YUV video sequence
H.265/HEVC DST/DCT Quantization
+
H.265/HEVC Entropy coding (CABAC)
H.265/HEVC video stream with payload message

Figure 11. Overall structure of the proposed data embedding method.

Algorithm 2: Embedding

```
    Data: Video sequence, input message, parameter $\kappa$
    Result: HEVC/H.265 compressed video with embedded message
 1  Convert input message to binary data matrix $D$ (e.g., using ASCII values if it is text) ;
 2  Encode data matrix $D$ with either BCH (Table 1) or Turbo coding scheme (Table 2). Let encoded
      data be $D_e$ ;
 3  Start encoding YUV video to H.265/HEVC;
 4  For each I-frame select the $4 \times 4$, $8 \times 8$ and $16 \times 16$ TU blocks ;
 5  Evaluate coarseness $C_{JSD-SM}$ of each $8 \times 8$ and $16 \times 16$ blocks using Algorithm 1 ;
 6  Sort two groups of $8 \times 8$ and $16 \times 16$ blocks based on their $C_{JSD-SM}$ values ;
 7  Select top $\kappa$ % most coarse blocks in each group of $8 \times 8$ and $16 \times 16$ TUs ;
 8  Embed encoded data $D_e$ sequentially in quantised DST coefficients of *all* $4 \times 4$ TUs using
      $SME(1,3,7)$ embedding technique as described in Section 5.2 ;
 9  Embed rest of the data in quantised DCT coefficients of *selected* $8 \times 8$ and $16 \times 16$ blocks using
      $SME(1,3,7)$ embedding technique as described in Section 5.2 ;
10  Continue HEVC compression process and the modified TU blocks are entropy coded (CABAC) ;
11  HEVC/H.265 compressed video stream with embedded payload data is output for storage or
      transmission ;
12  Output the *key* $K_{BCH} = \{n, k, t, \kappa\}$, if BCH error correcting code is used, where $n, k, t$ are
      parameters of BCH coding OR if Turbo coding is used, output key $K_{Turbo} = \{seq, k_t, \kappa\}$, where
      $seq = \{3, 1, 0, 3\}$ or $seq = \{2, 6, 4, 5, 1, 3\}$ (See Section 5.3) ;
```

Algorithm 3: Extraction

```
   Data: H.265/HEVC compressed video with embedded data, key
   Result: Extracted payload data
 1 Begin entropy decoding of the compressed video ;
 2 Extract all $4 \times 4$, $8 \times 8$ and $16 \times 16$ luma TUs from I-frames ;
 3 Measure JSD-SM block coarseness $C_{JSD-SM}$ of all $8 \times 8$ and $16 \times 16$ luma blocks using
     Algorithm 1 in spatial domain ;
 4 Sort the two groups of $8 \times 8$ and $16 \times 16$ blocks on the basis of $C_{JSD-SM}$ value ;
 5 Select $\kappa$ % most coarse blocks in each group of $8 \times 8$ and $16 \times 16$ luma TUs ;
 6 Extract embedded bits from all $4 \times 4$ and selected $8 \times 8$ and $16 \times 16$ luma TU blocks using
     SME(1,3,7) extraction process as described in Section 5.2 ;
 7 Combine all data to get the BCH or Turbo encoded data $D'_e$ that possibly contains some errors
     due to unreliable transmission or re-compression attack ;
 8 If $D'_e$ is BCH encoded, decode using Berelkamp's iterative algorithm [44] ;
 9 If $D'_e$ is Turbo encoded, decode with Soft-Output Viterbi algorithm (SOVA) [45] ;
10 Decoded data $D'$ is output as the extracted data ;
```

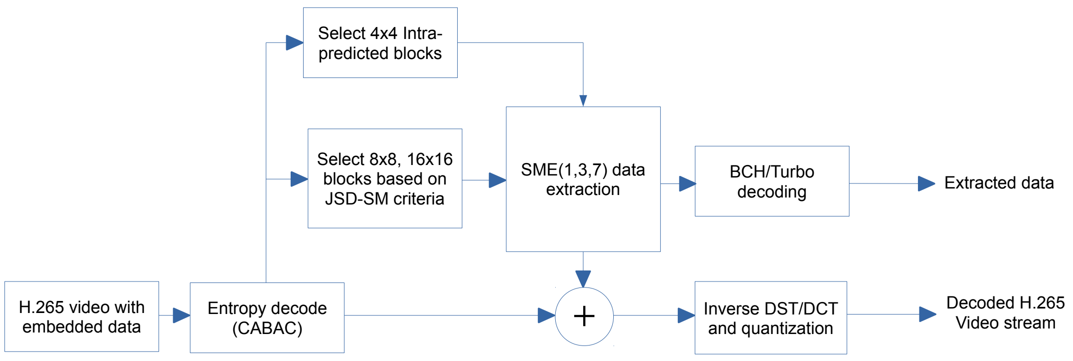
Select 4x4 Intra-predicted blocks
Select 8x8, 16x16 blocks based on JSD-SM criteria
SME(1,3,7) data extraction
BCH/Turbo decoding
Extracted data
H.265 video with embedded data
Entropy decode (CABAC)
+
Inverse DST/DCT and quantization
Decoded H.265 Video stream

Figure 12. Overall structure of the proposed data extraction method.

6. Experimental Results and Discussion

The proposed method was implemented in the H.265/HEVC reference coding software HM (version 16.20) released by Fraunhofer HH Institute. All experiments were run in a GNU/Linux based operating system on a computer with Intel i7-3770 CPU and 16GB RAM. Six test video sequences were used as the carrier videos. The names and specifications of these video sequences are given the Table 3. These videos were chosen because these are widely used by researchers in data hiding, video compression and allied fields and facilitate objective comparison of performances with other state-of-the-art data hiding methods proposed in the literature [14–16].

Table 3. List of test video sequences and their specifications.

Name	Resolution	RAW Format	HEVC GOP Length	GOP Conguration
Container	$352 \times 288\,(CIF)$	YUV	10	IBPBPBPBPB
News	$352 \times 288\,(CIF)$	YUV	10	IBPBPBPBPB
Mobile	$352 \times 288\,(CIF)$	YUV	10	IBPBPBPBPB
Akiyo	$352 \times 288\,(CIF)$	YUV	10	IBPBPBPBPB
Coastguard	$352 \times 288\,(CIF)$	YUV	10	IBPBPBPBPB
Foreman	$352 \times 288\,(CIF)$	YUV	10	IBPBPBPBPB

For objective comparison with the above mentioned research works, we used the *main* profile of the H.265/HEVC standard for video encoding purpose.

6.1. Visual Quality and Payload Capacity

For measuring and comparing the distortion characteristics of the proposed method, we used Peak Signal to Noise Ratio (*PSNR)*.PSNR is defined in terms of mean squared error (MSE). Given a $m \times n$ original image or video frame I and its corresponding stego-frame K, MSE is defined as:

$$MSE = \frac{1}{m\,n} \sum_{i=0}^{m-1} \sum_{j=0}^{n-1} \left[I(i,j) - K(i,j) \right]^2 \tag{25}$$

Then, $PSNR$ (in dB) is defined as:

$$PSNR = 10 \cdot \log_{10} \left(\frac{MAX_I^2}{MSE} \right) \tag{26}$$

where MAX is the maximum pixel value of the image/video. In our case, as the bit depth of the YUV test videos is 8, $MAX = 2^8 - 1$ = 255.

For objective comparison of our results with state-of-the-art in the literature, we selected the works discussed in [4,14]. These papers were selected as they are recent state-of-the-art and reported much better results than earlier works in the literature. Moreover, the researchers in these papers used the same set of test video sequences at the same resolution as in our work. For comparison of capacity and distortion performances, test video sequences must be same. These observations motivated us to compare our results with the results in these papers. We kept the quantisation parameter value (QP) fixed at 30. As in these studies, we encoded 300 frames of each test video sequences with a frame rate of 30 frames per second. We used the main profile of the H.265/HEVC encoder. Table 4 summarises the visual quality performances of all the proposed schemes in terms of PSNR values. In this table, PSNR$_1$ is the visual quality difference between the original YUV videos and the re-quantised video with $QP = 30$, but without embedded data. PSNR$_2$ values denote the visual quality difference between the original YUV videos and the re-quantised video (with $QP = 30$) with data embedded using the proposed method. Therefore, the quantity $(PSNR_1 - PSNR_2)$ denotes the loss in visual quality due to embedding. It is seen in the table that the maximum difference between PSNR$_1$ and PSNR$_2$ is less than 2 for all video sequences and proposed schemes. This indicates that visual distortion due to data embedding is hardly noticeable in naked human eye. This can be observed visually in Figure 13. Figure 13a,d,g,j shows the original frames from the original YUV video sequences container, news, mobile and akiyo, from top to bottom. The frames in the middle column are the corresponding frames from HEVC video sequences, compressed with $QP = 30$, but with no embedded data. The rightmost column shows the frames from the HEVC compressed video sequences with data embedded with the proposed method. It is seen that these frames are visually almost indistinguishable from the middle and leftmost frames. For closer inspection, we present enlarged segments of I-frames of video sequences in Figure 14. The left column shows the first HEVC compressed (with $QP = 30$) I-frames of Container, News, Mobile and Akiyo datasets from top to bottom, with no data embedded. The middle column shows the corresponding segments with data embedded. The right hand side column shows the matrices of absolute value differences between the pixels of segments with and without embedded data. It is seen that the difference matrices are almost black, indicating pixel value differences very close to zero. This vindicates the efficacy of the proposed method in preserving good visual quality.

Table 4. The PSNR values attained at with different coding schemes of the proposed method. PSNR_1 denotes the measured quality difference between the original YUV video sequences and the HEVC encoded video sequences with $QP = 30$, but no data are embedded. PSNR_2 values represent the attained quality when data are embedded in HEVC compressed video with $QP = 30$.

Video	**PSNR** $_1(QP = 30)$	**PSNR** $_2(QP = 30)$ **and SME**$(1, 3, 7)$ **and Following Error Codes**				
		B(7,4,1)	**B(31,16,3)**	**B(31,11,5)**	**T** $(K_t = 16)$	**T** $(K_t = 24)$
Container	39.41	38.25	38.37	38.28	38.06	38.22
News	38.96	38.00	37.81	37.93	37.90	37.94
Mobile	39.82	38.99	38.98	38.76	38.53	38.71
Akiyo	39.75	38.05	37.99	38.20	38.02	38.18
Coastguard	39.83	38.31	38.26	38.33	37.97	38.01
Foreman	39.47	38.73	38.28	38.36	38.53	38.24

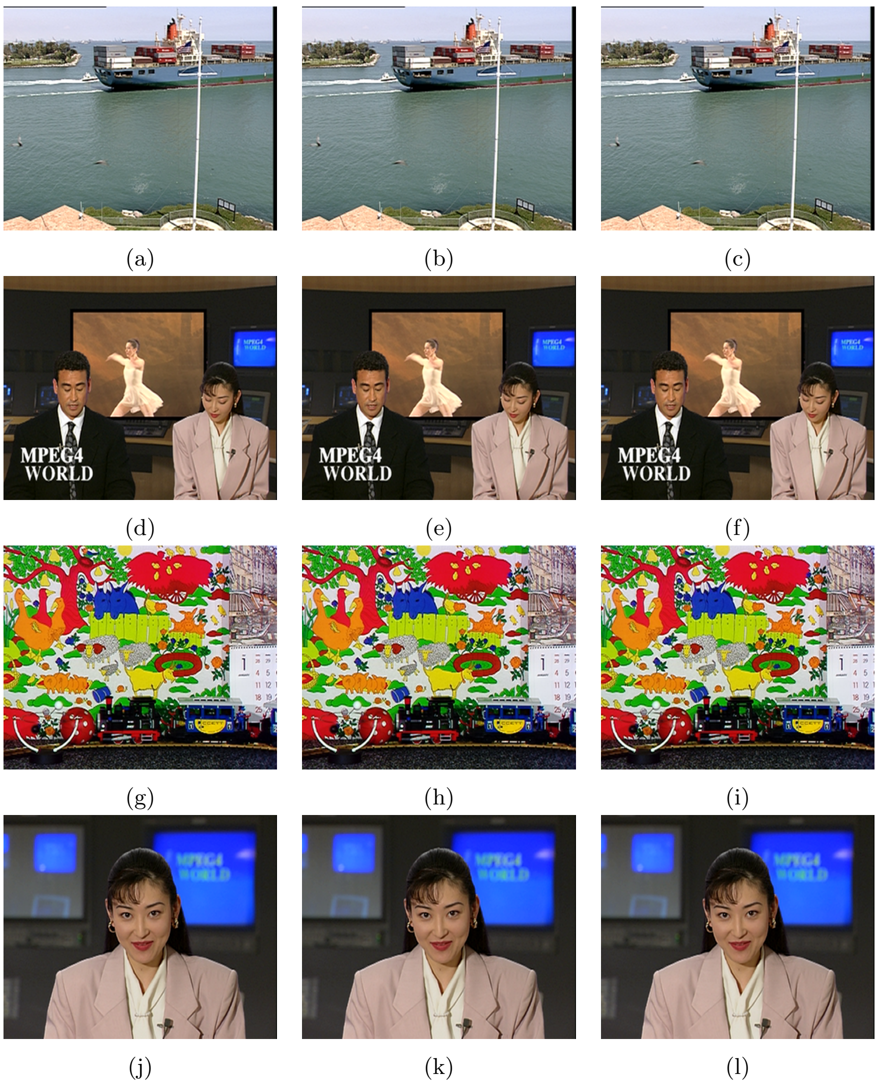

Figure 13. Frames on the leftmost column, i.e., (**a**,**d**,**g**,**j**) are the original frames from the original YUV video sequences container, news, mobile and akiyo, from top to bottom. The frames in the middle column (**b**,**e**,**h**,**k**) are the corresponding frames from HEVC video sequences, compressed with $QP = 30$, but with no embedded data. The rightmost column (**c**,**f**,**i**,**l**) shows the frames from the HEVC compressed video sequences with data embedded with the proposed method.

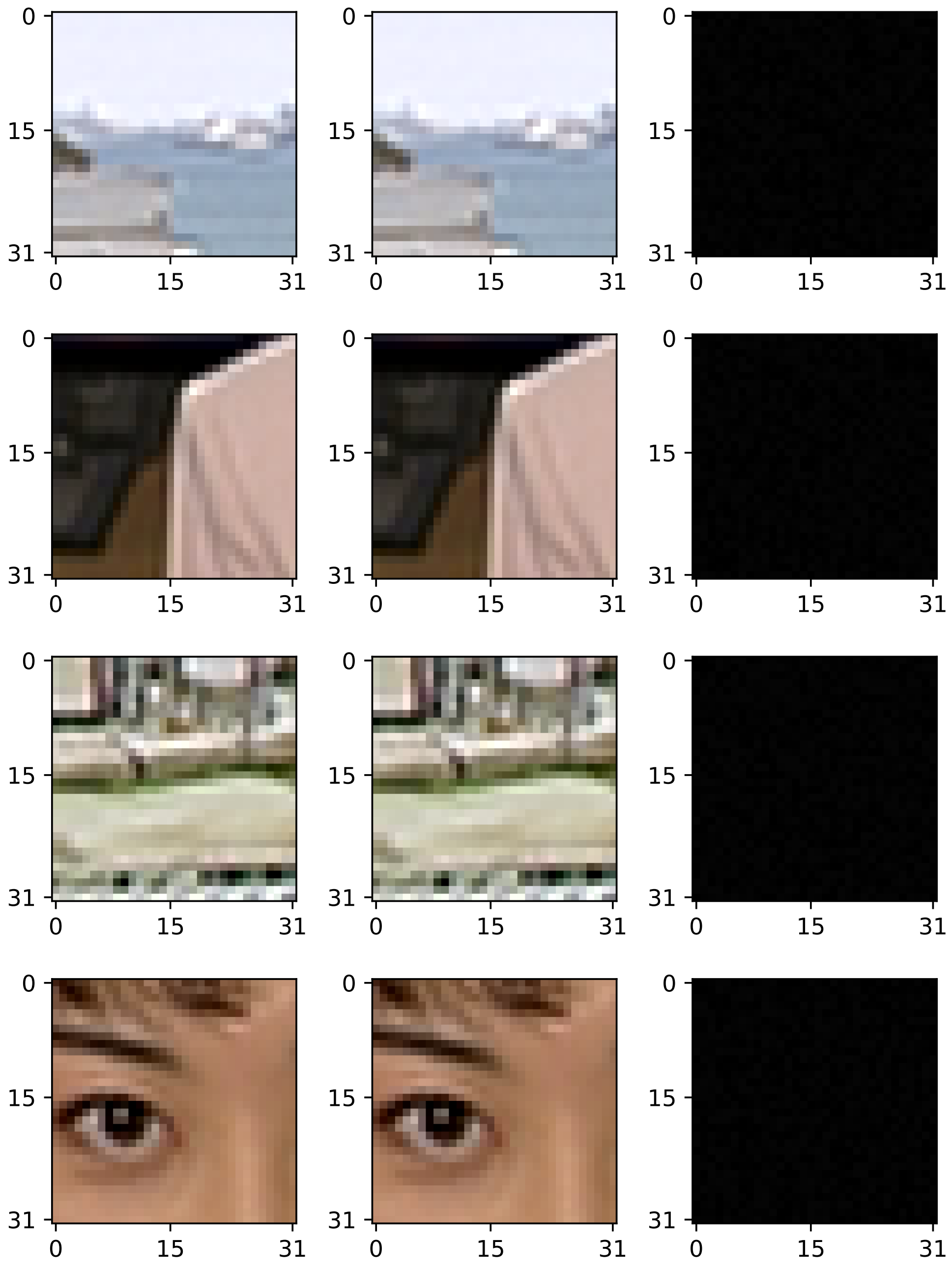

Figure 14. Enlarged 32 × 32 segments on the left column are from the first I-frames of the HEVC compressed (with $QP = 30$) video sequences Container, News, Mobile and Akiyo, from top to bottom. The middle column shows the corresponding enlarged frames with embedded data. The right hand column shows the absolute value difference matrices between the segments with and without embedded data.

Table 5 presents the comparison of PSNR values and attained payload capacity between the proposed method and the aforementioned state-of-the-art methods in teh literature. Table 5 shows that the proposed method attains much greater payload capacity than the methods of Liu et al. [4] and Liu et al. [14]. Liu et al. [4] proposed multiple schemes of embedding. Among those, the highest performing scheme was compared. In the proposed method, the SME(1,3,7) embedding scheme together with the BCH(7,4,1) error correcting code yields PSNR more than 38 and capacity of over 400 bits on the average per I-frame for all the test video sequences. Therefore, average payload capacity is three times than that of the method in [14] and more than 2.5 times the capacity offered by the method in [4]. In the proposed method, when SME(1,3,7) embedding together with Turbo($K_t = 24$) error correcting scheme is used, the embedding capacity is more than twice the capacity of the method in [4] and almost two times the method in [14]. Despite drastic increase of capacity, the proposed method does not compromise with visual quality. In all cases, the proposed schemes attain slightly better PSNR values than the methods of both Liu et al. [4] and Liu et al. [14]. Early methods in the literature selected only the 4×4 TU blocks in order to keep the visual quality degradation under control. This, however, yielded low payload capacity. In the proposed method, in addition to all 4×4 TU blocks, some 8×8 and 16×16 TU blocks that pass the proposed JSD-SM coarseness criteria, are also used for data embedding venue. This drastically increases payload capacity. Moreover, the SME$(1, 3, 7)$ data modulation technique used in this paper is able to embed 3 bits of data in a block of 7 QTCs by changing *only* 1 of the QTC values. Thus, SME(1,3,7) results in less change density, i.e., fewer modifications per embedded bit. Fewer changes of QTCs gives more headroom to embed data in blocks larger than 4×4. Hence, the proposed method is able to embed data in 8×8 and 16×16 blocks without negatively affecting visual quality of stego-video.

Table 5. Comparison of visual quality (PSNR) and payload capacity between various schemes of the proposed method and in [4,14]. PSNR is in dB and Capacity is given in bits. As Liu et al. [4] proposed many error correcting schemes, only the best performing scheme was compared.

Video	**Liu et al. [4]**		**Liu et al. [14]**		**SME(1,3,7) + BCH(7,4,1)**		**SME(1,3,7) + Turbo($K_t = 24$)**	
	PSNR	**Capacity**	**PSNR**	**Capacity**	**PSNR**	**Capacity**	**PSNR**	**Capacity**
Container	36.50	171	36.44	120	38.25	412	38.22	338
News	36.50	171	36.95	132	38.00	434	37.94	346
Mobile	36.50	171	34.24	172	38.99	518	38.71	392
Akiyo	36.50	171	38.63	124	38.05	446	38.18	354
Coastguard	36.50	171	–	–	38.31	438	39.01	348
Foreman	36.50	171	–	–	38.73	422	38.24	336

6.2. Robustness Performance

For robustness analysis of the proposed framework, we adopted the following methodology. First, we embeded data in the video sequences using different error correcting coding schemes proposed in Section 5.3. Next, the video sequences with embedded data were re-quantised with different values of quantisation parameter (QP). Then, the payload data were extracted from the re-quantised video sequences. Recovered payload data were then compared to the original data and bit-by-bit similarity was evaluated. The higher is the similarity, the higher is the survivability or robustness. For the the purpose of objective comparison with the methods of Swati et al. [15] and Liu et al. [14], the QP values are taken same as in those methods, i.e., from $QP = 29$ to $QP = 35$ with increment step of 1.

The results are summarised in Table 6. As seen in the table, in case of the Container video, survival rate attained by proposed BCH$(7, 4, 1)$ error correcting code is better than both Swati et al. [15] and Liu et al. [14] at all QP values. As expected, BCH$(31, 11, 5)$ scheme performs much better than both the literature at all QP values in the range 29–35. The proposed Turbo error correcting schemes gives interesting results. As seen is the table, the coding scheme with block length $K_t = 16$ performs much better than both Swati et al. [15] and Liu et al. [14]. In addition, its performance is similar to BCH(31,11,5). This means that, in the proposed method, Turbo code with block length 16 is almost as powerful as BCH(31,11,5). The fact that Turbo code is able to correct similar number of errors with smaller block size can be attributed to its superior burst error correcting capability with the help of the parallel concatenated recursive convolution codes and the pseudo-random interleaver (Section 4.2). In the case of the News and Mobile video sequences, similar superior performance of Turbo code is noticed.

Table 6. Survival rate of the proposed method at different QP values when BCH(7,4,1), BCH(31,16,3), BCH(31,11,5), Turbo$(K_t = 16)$ and Turbo$(K_t = 124)$ error correcting codes are used. Performances of these codes have been compared to the performance of the methods of Swati et al. [15] and Liu et al. [14].

Video	**QP**	**[15]**	**[14]**	**BCH(7,4,1)**	**BCH(31,16,3)**	**BCH(31,11,5)**	**Turbo(16)**	**Turbo(24)**
Container	29	63%	61%	74.22%	81.01%	95.10%	96.00%	96.10%
	30	65%	78%	89.04%	97.07%	99.02%	98.56%	99.72%
	31	78%	80%	91.17%	98.99%	100%	100%	100%
	32	87%	83%	92.50%	100%	100%	100%	100%
	33	61%	85%	94.50%	99%	100%	100%	100%
	34	61%	54%	76.66%	88.45%	94.05%	95.44%	98.29%
	35	42%	22%	64.53%	79.10%	83.87%	89.39%	91.44%
News	29	60%	59%	82.41%	92.15%	94.61%	96.20%	96.35%
	30	63%	82%	88.23%	97.82%	99.98%	99.99%	99.99%
	31	56%	80%	90.84%	99.95%	100%	100%	100%
	32	70%	81%	89.90%	100%	100%	100%	100%
	33	46%	85%	91.92%	99.98%	100%	100%	100%
	34	44%	38%	76.61%	91.40%	93.00%	95.16%	96.60%
	35	42%	31%	63.96%	84.35%	88.55%	91.25%	92.91%
Mobile	29	59%	60%	80.95%	92.18%	94.61%	97.01%	98.77%
	30	63%	77%	85.77%	97.83%	99.24%	99.89%	99.98%
	31	65%	82%	90.50%	98.59%	100%	100%	100%
	32	71%	85%	92.94%	100%	100%	100%	100%
	33	67%	68%	91.97%	98.00%	100%	100%	100%
	34	59%	59%	77.90%	92.49%	93.28%	95.10%	96.62%
	35	51%	44%	69.09%	87.35%	88.94%	92.56%	92.99%

Robustness performance of the proposed method was also evaluated in terms of similarity (SIM) and bit error error rate (BER). For original embedded data $D(i,j)$ and recovered data $\hat{D}(i,j)$ of size $m \times n$, SIM and BER are defined as follows.

$$SIM = \frac{\sum_{i=1}^{m}\sum_{j=1}^{n}\left[D(i,j) \times \hat{D}(i,j)\right]}{\sqrt{\sum_{i=1}^{m}\sum_{j=1}^{n} D(i,j)^2} \times \sqrt{\sum_{i=1}^{m}\sum_{j=1}^{n} \hat{D}(i,j)^2}} \tag{27}$$

$$BER = \frac{\sum_{i=1}^{m}\sum_{j=1}^{n}\left[D(i,j) \oplus \hat{D}(i,j)\right]}{m \times n} \tag{28}$$

The results are compared and summarised in Table 7. In the table, it can be seen that the average similarity values achieved by the proposed framework are much higher than those of Swati et al. [15] and Liu et al. [14] and is slightly better than those of Liu et al. [16] at all QP values. Here, it is to be noted that Liu et al. [14] used same set of test video sequences as used in this work. However, both Swati et al. [15] and Liu et al. [16] used High resolution video sequences such as ParkScene (1920 × 1080), BQMall (832 × 480), PeopleOnStreet (2560 × 1600), etc. which used to be available from University of Hannover [46], but are no longer accessible, and therefore could not be evaluated in our work. The main difference between these videos and the videos used in our work is that these videos are very high resolution whereas our test sequences are of resolution 352 × 288. Therefore, comparison of our results with these works is not perfect, but since both BER and SIM are given in percentage (i.e., a dimensionless quantity), it gives a fair idea as to where our results stands in comparison to Swati et al. [15] and Liu et al. [16].

Table 7. Comparison of average SIM and BER robustness values of the proposed method with that of the methods proposed by Liu et al. [16], Swati et al. [15] and Liu et al. [14].

Attack Type		**Liu et al. [16]**		**Swati et al. [15]**		**Liu et al. [14]**		**Proposed (avg)**	
		SIM	**BER**	**SIM**	**BER**	**SIM**	**BER**	**SIM**	**BER**
No attack		1	0	1	0	1	0	1	0
Re-quantisation attack	QP = 30	0.94	6	0.62	38.35	0.80	22.40	0.98	2
	QP = 31	1	0	0.80	22.34	0.81	20.70	1	0
	QP = 32	1	0	0.86	15.60	0.85	17.20	1	0
	Qp = 33	0.96	4	0.58	42.50	0.85	16.30	0.98	2
	QP = 34	0.80	22	0.58	42.20	0.50	45.98	0.88	11

6.3. Bit-Rate Increase

The bit-rate (and consequently total size of video sequence) tends to increase after data embedding. The percentage of this increment should be as low as possible. Bit-rate increase due to data embedding is measured as follows. First, the original YUV video sequences were compressed in HEVC with $QP = 30$, but no data were embedded. Next, the YUV sequences were compressed in HEVC with $QP = 30$ and data were embedded using the proposed method. The percentage of increase of size (in bits) is the increase in bit-rate. For example, size of the HEVC compressed Container sequence (when no data are embedded) is 9,299,832 bytes. When Turbo coded data (with $K_t = 24$) is embedded using the proposed method, size increases to 9,671,825. This is a 0.04% increase of bit-rate. Table 8 summarises the results and compares to Liu et al. [14]. The table shows that the proposed method causes some increase in bit-rate that is lower than that of Liu et al. [14]. In [14], Liu et al. used same set of test video sequences as used in this work, making direct comparison possible. However, other recent state-of-the-art techniques such as those of Liu et al. [16] and Li et al. [47] used a set of higher resolution video sequences that makes these techniques incompatible for direct comparison with the proposed method. Nevertheless, the average bit-rate increase in these works and our proposed work are presented in Table 9 together with the specifications of the video dataset used.

Table 8. Comparison of video bit-rate increase due to data embedding. All values are percentages.

Video	**Liu et al. [14]**	**Proposed Method (SME with Following Error Codes)**				
		BCH(7,4,1)	**BCH(31,16,3)**	**BCH(31,11,5)**	**Turbo(16)**	**Turbo(24)**
Container	2.7	0.01	0.02	0.02	0.03	0.04
News	2.8	0.02	0.02	0.03	0.03	0.05
Mobile	4.0	0.02	0.03	0.02	0.04	0.05
Akiyo	1.4	0.02	0.03	0.02	0.04	0.05
Coastguard	-	0.02	0.03	0.03	0.04	0.06
Foreman	-	0.01	0.03	0.04	0.04	0.06

Table 9. Comparison of average bit-rate increment of the proposed work with that of recent state-of-art techniques that used other video datasets.

Technique	**Video Dataset**	**Bit Rate Increase (avg.)**
Liu et al. [16]	ParkScene (1920 × 1080), FourPeople (1280 × 720), KirstenAndSara (1280 × 720), etc. PartyScene (832 × 480), BQMall (832 × 480), RaceHorses (416 × 240)	0.785%
Li et al. [47]	BasketBallDrive (1920 × 1080), ParkScene (1920 × 1080), BQTerrace (1920 × 1080), Kimono (1920 × 1080), ChinaSpeed (1024 × 768), Keiba (832 × 480), etc.	0.014%
Proposed	Container (352 × 288), News (352 × 288), Mobile (352 × 288), Akiyo (352 × 288), Coastguard (352 × 288) and Foreman (352 × 288)	0.031%

6.4. JSD-SM Coarness Measure Analysis

In the proposed data hiding method, JSD-SM coarseness measure (Algorithm 1) plays a crucial role. As the theory of Human Visual System (HVS) [48] suggests that the human vision is less sensitive to changes in the luminosity in coarse regions, the aim is to embed data in relatively coarse TU blocks of the luma channel of each I-frame in the video sequence. The HEVC compression algorithm allocates 4×4 TU blocks to the most coarse areas. Therefore, in this paper, data are embedded in all 4×4 blocks. To achieve greater payload capacity, data should be embedded in larger TU blocks. However, this may cause unacceptable visual distortion. Hence, only relatively coarser blocks among the 8×8 and 16×16 luma TU blocks are selected as embedding venues. The top κ% most uneven 8×8 and 16×16 luma TU blocks are selected based on the proposed JSD-SM coarseness measure, as explained in Section 5.1. Data are embedded in these selected blocks using the SME(1,3,7) technique. Figure 15 illustrates four specimen 16×16 luma TU blocks from the second I-frame of the Akiyo video sequence. The top left block is a non selected block as it contains little variation of luminosity. Other three blocks were selected as these has high coarseness as per JSD-SM coarseness measure. These show that the proposed JSD-SM coarseness measure correctly captures the relatively more coarse blocks. The quantity κ works as a parameter to the embedding algorithm. If $\kappa = p$, the top p% most coarse 8×8 and 16×16 TU blocks are selected for data embedding. Thus, κ controls the balance between embedding capacity and embedding distortion. If $\kappa = 0$, none of the 8×8 and 16×16 blocks are selected for embedding and data are embedded only in the 4×4 blocks. Similarly, if $\kappa = 100$, all 8×8 and 16×16 blocks are selected as data embedding venues.

Figure 16 shows the variation of average embedding capacity in I-frames at different values of κ. As the parameter κ specifies the percentage of most coarse blocks to be selected, both payload capacity and embedding distortion are dependant on κ. In the figure, it can be seen that at $\kappa = 0$, i.e., when data are embedded in only the 4×4 blocks and in no 8×8 or 16×16 blocks, the average capacity varies from 256 bits to 380 bits. The capacity of the Foreman video sequence is the least because among all test sequences, it contains least number of 4×4 blocks at QP = 30. On the other hand, the Mobile test sequence contains more coarseness, and it is allocated more 4×4 blocks at QP = 30. Hence, it has the highest capacity among all test videos at QP = 30. At $\kappa = 20$, in addition to all the 4×4 blocks, data are embedded in the 20% most coarse 8×8 and 16×16 blocks that are selected based on the proposed JSD-SM block coarseness measure. Consequently, payload capacity increases significantly for all the test sequences. Similar to every increment of κ from 0 to 100, embedding capacity increases as more and more 8×8 and 16×16 are selected as embedding venue. At $\kappa = 100$, the highest capacity is achieved as data are embedded in all 8×8 and 16×16 blocks.

Figure 17 illustrates distortion performance in terms of PSNR at different values of κ at QP = 30. At $\kappa = 0$, i.e., when data are embedded only in the 4×4 blocks, highest PSNR values are achieved for all the test video sequences and these values range from 38 to 39. At $\kappa = 20$, in addition to the 4×4 blocks, 20% of the 8×8 and 16×16 blocks are used as data embedding venues and therefore, as more QTCs are modified, PSNR value drops. At $\kappa = 100$, *all* 4×4, 8×8 and 16×16 blocks in an I-frame are used as data embedding venues. Consequently, PSNR values drop even further. However, even at $\kappa = 100$, PSNR values stay above 35. These results prove the proposed method achieves good distortion characteristics. Moreover, the proposed method provides great flexibility of attainable payload capacity and distortion performance. It lets the user choose a value of κ that dictates the balance between payload capacity and distortion.

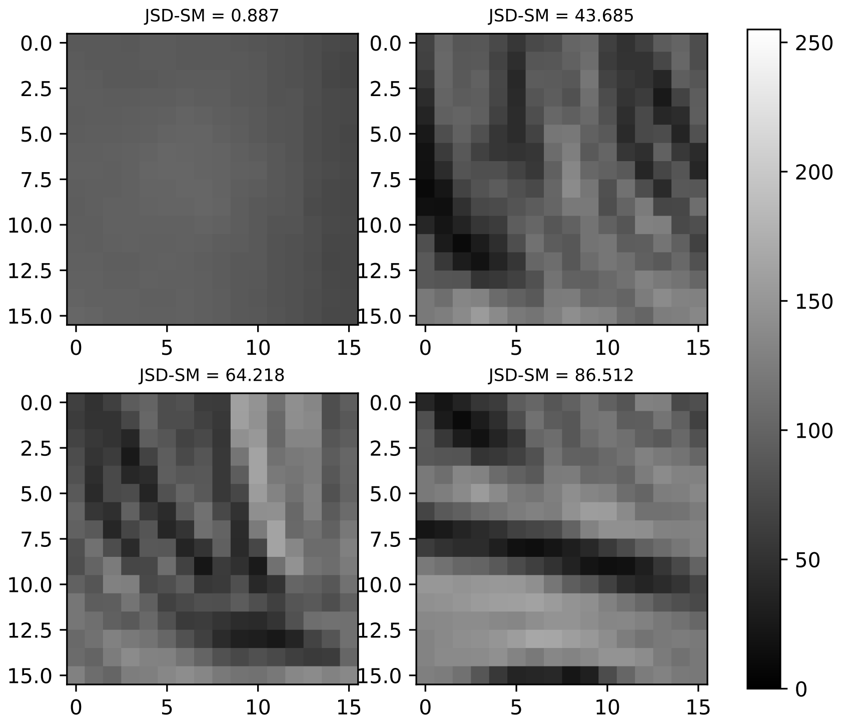

Figure 15. Four specimen 16 × 16 luma TU blocks from the second I-frame of the *Akiyo* video sequence. The top left block is a unselected block as it contains little variation of luminosity. Other three blocks were selected as these has high coarseness as per JSD-SM coarseness measure.

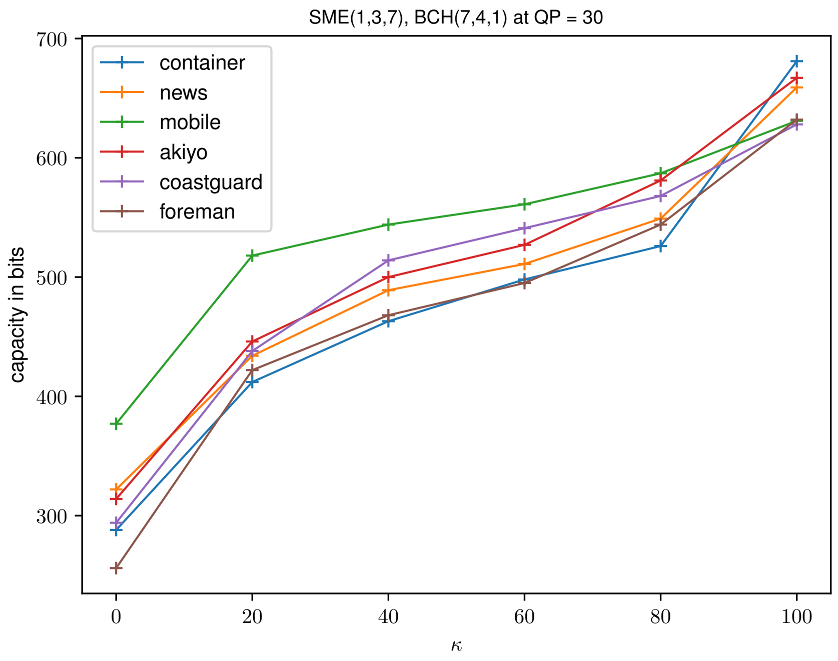

Figure 16. Data embedding capacity of the six video sequences at $QP = 30$ with different values of κ.

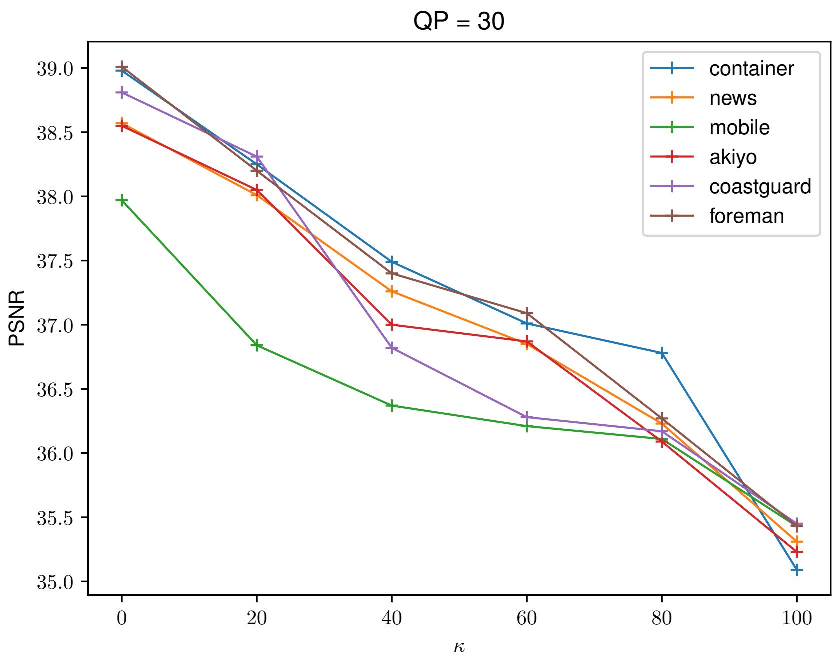

Figure 17. PSNR values attained with the six video sequences at different values of κ, when BCH(7,4,1) error correcting scheme is used in conjunction with SME(1,3,7) data embedding technique

6.5. Computation Time

The proposed method was implemented in HM (version 16.20), which is the H.265/HEVC reference coding software released by Fraunhofer HH Institute. All experiments were run in a GNU/Linux based operating system on a computer with Intel i7-3770 CPU and 16GB RAM. The HM software is a C++ a source-only package and researchers are supposed to modify the software as per need and then compile to generate the executable encoder, decoder and other modules. We compiled the software in GNU C++ compiler with optimiser switch "-O 2". This is important to mention, because an executable generated with different optimiser switch may give different execution time for experiment. For the sake of consistency, CPU throttling was turned off and the clock speed was kept fixed at 2.4 GHz. Multi-threading was also turned off as it may give dynamic behaviour in execution. The computation time of the proposed embedding method can be divided in the following steps.

A. Average data pre-processing and data encoding time: in this step, the data to be embedded are first converted to binary bit-stream from its original format. Then, they are encoded in one of the schemes of BCH or Turbo coding described in Section 5.3. Different schemes of BCH and Turbo encoding take slightly different time. The average time taken by all the proposed schemes is considered.
B. block selection using proposed JSD-SM technique as described in Section 5.1
C. data embedding using SME(1,3,7) technique as described in Section 5.2
D. total time: The total time taken to complete the whole embedding process. This includes A, B, C and rest of the usual HEVC process such as motion vector analysis, quantisation, entropy coding, etc.

Similarly, computation time for the data extraction process can be divided into following steps:

M. Block selection using the proposed JSD-SM technique
N. Data extraction using SME(1,3,7) technique
O. Data decoding using one of the proposed schemes of Turbo/BCH coding and post processing
P. Total time that includes M, N, O and rest of the HEVC decoding steps, e.g. inverse DST/DCT, quantisation, etc.

Tables 10 and 11 summarise the computation time of different steps and also the total time taken by embedding and extraction processes for each test video sequences. The results shown are the times taken to embed data in 300 frames of each video sequence.

Table 10. Computation time taken by the embedding process. All times are in seconds.

Name	A	B	C	D
Container	0.6	32.6	11.8	221.2
News	0.6	31.4	12.3	224.7
Mobile	0.6	33.8	11.9	229.0
Akiyo	0.6	33.7	12.1	219.8
Coastguard	0.6	32.2	12.3	224.2
Foreman	0.6	33.0	11.9	232.3

Table 11. Computation time of the extraction process. All times are in seconds.

Name	M	N	O	P
Container	31.4	13.2	1.3	102.8
News	32.6	12.8	1.4	105.1
Mobile	33.1	13.1	1.6	110.7
Akiyo	32.7	14.0	1.4	108.5
Coastguard	31.4	13.5	1.5	111.8
Foreman	32.3	13.2	1.6	118.0

7. Conclusions

In this paper, we propose a method of robust and secure data hiding in H.265/HEVC compressed videos that is capable of higher payload capacity than the previous state-of-the-art methods in the literature at similar or better visual distortion. Compared to the previous state-of-the-art works in the literature, the proposed method achieves two to three times more payload capacity without compromising the visual quality. Higher payload capacity has been achieved by embedding data in larger transform blocks of H.265/HEVC and embedding distortion is kept under control by reducing embedding change density with the help of $\text{SME}(1,3,7)$ data embedding technique. $\text{SME}(1,3,7)$ is capable of embedding 3 bits of data in a block of seven QTCs by modifying only one QTC. Visual distortion is also kept under control by avoiding data embedding in smoother 8×8 and 16×16 blocks. The smooth blocks are filtered out and coarse blocks are selected as embedding venue with the help of the proposed JSD-SM block coarseness measure that is based on Jensen–Shannon divergence and second moment of pixel value distribution in image blocks. Moreover, the proposed method achieves excellent robustness against re-quantisation attacks with the help of powerful BCH or Turbo error correcting codes. We have compared a set of different parameters and generator polynomials of BCH and Turbo codes. The results show that Turbo code can achieve superior robustness due to its burst error correcting properties. However, there is room for more improvements. A potential way forward is to increase payload capacity even more by embedding data in the inter-predicted frames (P and B) and other HEVC compression features such as motion vectors.

Author Contributions: Conceptualisation, K.B.; methodology, K.B.; software, K.B.; validation, K.B.; formal analysis, K.B.; investigation, K.B.; resources, K.B.; data curation, K.B.; writing–original draft preparation, K.B.; writing–review and editing, K.B.; visualisation, K.B.; funding acquisition, K.B.

Funding: This research was partially funded by CSIR, India vide grant id. 09/263(1045)/2015-EMR-I, and by DST PURSE, JNU.

Conflicts of Interest: The author declares no conflict of interest.

References

1. Ohm, J.R.; Sullivan, G.J.; Schwarz, H.; Tan, T.K.; Wiegand, T. Comparison of the coding efficiency of video coding standards—Including high efficiency video coding (HEVC). *IEEE Trans. Circuits Syst. Video Technol.* **2012**, *22*, 1669–1684. [CrossRef]
2. Chang, P.C.; Chung, K.L.; Chen, J.J.; Lin, C.H.; Lin, T.J. A DCT/DST-based error propagation-free data hiding algorithm for HEVC intra-coded frames. *J. Vis. Commun. Image Represent.* **2014**, *25*, 239–253. [CrossRef]
3. Liu, Y.; Li, Z.; Ma, X.; Liu, J. A robust data hiding algorithm for H.264/AVC video streams. *J. Syst. Softw.* **2013**, *86*, 2174–2183. [CrossRef]
4. Liu, Y.; Hu, M.; Ma, X.; Zhao, H. A new robust data hiding method for H.264/AVC without intra-frame distortion drift. *Neurocomputing* **2015**, *151*, 1076–1085. [CrossRef]
5. 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. Circuits Syst. Video Technol.* **2010**, *20*, 1320–1330. [CrossRef]
6. Xu, D.; Wang, R.; Shi, Y.Q. Data hiding in encrypted H. 264/AVC video streams by codeword substitution. *IEEE Trans. Inf. Forensics Secur.* **2014**, *9*, 596–606. [CrossRef]
7. Niu, K.; Yang, X.; Zhang, Y. A novel video reversible data hiding algorithm using motion vector for H. 264/AVC. *Tsinghua Sci. Technol.* **2017**, *22*, 489–498. [CrossRef]
8. Ma, Z.; Huang, J.; Jiang, M.; Niu, X. A video watermarking DRM method based on H. 264 compressed domain with low bit-rate increasement. *Chin. J. Electron.* **2016**, *25*, 641–647. [CrossRef]
9. Lin, T.J.; Chung, K.L.; Chang, P.C.; Huang, Y.H.; Liao, H.Y.M.; Fang, C.Y. An improved DCT-based perturbation scheme for high capacity data hiding in H.264/AVC intra frames. *J. Syst. Softw.* **2013**, *86*, 604–614. [CrossRef]
10. Stutz, T.; Autrusseau, F.; Uhl, A. Non-blind structure-preserving substitution watermarking of H. 264/CAVLC inter-frames. *IEEE Trans. Multimed.* **2014**, *16*, 1337–1349. [CrossRef]
11. Ogawa, K.; Ohtake, G. Watermarking for HEVC/H.265 stream. In Proceedings of the 2015 IEEE International Conference on Consumer Electronics (ICCE), Las Vegas, NV, USA, 9–12 January 2015; pp. 102–103. [CrossRef]
12. Liu, Y.; Liu, S.; Zhao, H.; Liu, S. A new data hiding method for H.265/HEVC video streams without intra-frame distortion drift. *Multimed. Tools Appl.* **2018**. [CrossRef]
13. Dutta, T.; Gupta, H.P. A robust watermarking framework for High Efficiency Video Coding (HEVC)—Encoded video with blind extraction process. *J. Vis. Commun. Image Represent.* **2016**, *38*, 29–44. [CrossRef]
14. Liu, Y.; Chen, L.; Hu, M.; Jia, Z.; Jia, S.; Zhao, H. A reversible data hiding method for H. 264 with Shamir's (t, n)-threshold secret sharing. *Neurocomputing* **2016**, *188*, 63–70. [CrossRef]
15. Swati, S.; Hayat, K.; Shahid, Z. A watermarking scheme for high efficiency video coding (HEVC). *PLoS ONE* **2014**, *9*, e105613. [CrossRef]
16. Liu, Y.; Zhao, H.; Liu, S.; Feng, C.; Liu, S. A robust and improved visual quality data hiding method for HEVC. *IEEE Access* **2018**, *6*, 53984–53997. [CrossRef]
17. Gaj, S.; Sur, A.; Bora, P.K. A robust watermarking scheme against re-compression attack for H. 265/HEVC. In Proceedings of the 2015 Fifth National Conference on Computer Vision, Pattern Recognition, Image Processing and Graphics (NCVPRIPG), Patna, India, 16–19 December 2015; pp. 1–4.
18. Tew, Y.; Wong, K. Information hiding in HEVC standard using adaptive coding block size decision. In Proceedings of the 2014 IEEE International Conference on Image Processing (ICIP), Paris, France, 27–30 October 2014; pp. 5502–5506.
19. Yang, Y.; Li, Z.; Xie, W.; Zhang, Z. High capacity and multilevel information hiding algorithm based on pu partition modes for HEVC videos. *Multimed. Tools Appl.* **2019**, *78*, 8423–8446. [CrossRef]
20. Bo, P.; Jie, Y. A Reversible Information Hiding Method Based on HEVC. *IFAC-PapersOnLine* **2018**, *51*, 238–243. [CrossRef]
21. Thiesse, J.M.; Jung, J.; Antonini, M. Rate distortion data hiding of motion vector competition information in chroma and luma samples for video compression. *IEEE Trans. Circuits Syst. Video Technol.* **2011**, *21*, 729–741. [CrossRef]
22. Aly, H.A. Data hiding in motion vectors of compressed video based on their associated prediction error. *IEEE Trans. Inf. Forensics Secur.* **2010**, *6*, 14–18. [CrossRef]

23. Sullivan, G.J.; Ohm, J.R.; Han, W.J.; Wiegand, T. Overview of the high efficiency video coding (HEVC) standard. *IEEE Trans. Circuits Syst. Video Technol.* **2012**, *22*, 1649–1668. [CrossRef]
24. Sole, J.; Joshi, R.; Nguyen, N.; Ji, T.; Karczewicz, M.; Clare, G.; Henry, F.; Duenas, A. Transform coefficient coding in HEVC. *IEEE Trans. Circuits Syst. Video Technol.* **2012**, *22*, 1765–1777. [CrossRef]
25. Sze, V.; Budagavi, M.; Sullivan, G.J. High efficiency video coding (HEVC). *Integr. Circuit Syst. Algorithms Archit. Springer* **2014**, *39*, 40.
26. Pastuszak, G. Flexible architecture design for H. 265/HEVC inverse transform. *Circuits Syst. Signal Process.* **2015**, *34*, 1931–1945. [CrossRef]
27. Wiegand, T.; Sullivan, G.J.; Bjontegaard, G.; Luthra, A. Overview of the H. 264/AVC video coding standard. *IEEE Trans. Circuits Syst. Video Technol.* **2003**, *13*, 560–576. [CrossRef]
28. Marpe, D.; Schwarz, H.; Wiegand, T. Context-based adaptive binary arithmetic coding in the H. 264/AVC video compression standard. *IEEE Trans. Circuits Syst. Video Technol.* **2003**, *13*, 620–636. [CrossRef]
29. Liu, Y.; Li, Z.; Ma, X.; Liu, J. A robust without intra-frame distortion drift data hiding algorithm based on H. 264/AVC. *Multimed. Tools Appl.* **2014**, *72*, 613–636. [CrossRef]
30. Mstafa, R.J.; Elleithy, K.M. A high payload video steganography algorithm in DWT domain based on BCH codes (15, 11). In Proceedings of the 2015 Wireless Telecommunications Symposium (WTS), New York, NY, USA, 15–17 April 2015; pp. 1–8.
31. Mstafa, R.J.; Elleithy, K.M. A DCT-based robust video steganographic method using BCH error correcting codes. In Proceedings of the 2016 IEEE Long Island Systems, Applications and Technology Conference (LISAT), Farmingdale, NY, USA, 29 April 2016; pp. 1–6.
32. Yoo, H.; Jung, J.; Jo, J.; Park, I.C. Area-efficient multimode encoding architecture for long BCH codes. *IEEE Trans. Circuits Syst. II Express Briefs* **2013**, *60*, 872–876. [CrossRef]
33. Hagenauer, J.; Offer, E.; Papke, L. Iterative decoding of binary block and convolutional codes. *IEEE Trans. Inf. Theory* **1996**, *42*, 429–445. [CrossRef]
34. Massey, J. Shift-register synthesis and BCH decoding. *IEEE Trans. Inf. Theory* **1969**, *15*, 122–127. [CrossRef]
35. Berrou, C.; Glavieux, A.; Thitimajshima, P. Near Shannon limit error-correcting coding and decoding: Turbo-codes. 1. In Proceedings of the ICC'93—IEEE International Conference on Communications, Geneva, Switzerland, 23–26 May 1993; Volume 2, pp. 1064–1070. [CrossRef]
36. Endres, D.M.; Schindelin, J.E. A new metric for probability distributions. *IEEE Trans. Inf. Theory* **2003**, *49*, 1858–1860 . [CrossRef]
37. Chan, C.K.; Cheng, L. Hiding data in images by simple {LSB} substitution. *Pattern Recognit.* **2004**, *37*, 469–474. [CrossRef]
38. Li, B.; He, J.; Huang, J.; Shi, Y.Q. A survey on image steganography and steganalysis. *J. Inf. Hiding Multimed. Signal Process.* **2011**, *2*, 142–172.
39. Wang, R.Z.; Lin, C.F.; Lin, J.C. Image hiding by optimal LSB substitution and genetic algorithm. *Pattern Recognit.* **2001**, *34*, 671–683. [CrossRef]
40. Roque, J.J.; Minguet, J.M. SLSB: Improving the Steganographic Algorithm LSB. In Proceedings of the Workshop on Security in Information Systems (WOSIS 2009), Milan, Italy, 6–7 May 2009; pp. 57–66.
41. Gutub, A.A.A. Pixel indicator technique for RGB image steganography. *J. Emerg. Technol. Web Intell.* **2010**, *2*, 56–64. [CrossRef]
42. Fridrich, J.; Lisoněk, P.; Soukal, D. On steganographic embedding efficiency. In Proceedings of the International Workshop on Information Hiding, Alexandria, VA, USA, 10–12 July 2006; Springer: Berlin/Heidelberg, Germany, 2006; pp. 282–296.
43. Fridrich, J.; Soukal, D. Matrix embedding for large payloads. *IEEE Trans. Inf. Forensics Secur.* **2006**, *1*, 390–395. [CrossRef]
44. Berlekamp, E.R. *Non-Binary BCH Decoding*; Technical Report; Department of Statistics, North Carolina State University: Raleigh, NC, USA, 1966.
45. Hagenauer, J.; Hoeher, P. A Viterbi algorithm with soft-decision outputs and its applications. In Proceedings of the 1989 IEEE Global Telecommunications Conference and Exhibition'Communications Technology for the 1990s and Beyond', Dallas, TX, USA, 27–30 November 1989; pp. 1680–1686.
46. University of Hannover. High Resolution Video Datasets. 2013. Available online: http://ftp.tnt.uni-hannover.de/testsequences (accessed on 7 July 2019).

47. Li, Z.; Meng, L.; Jiang, X.; Li, Z. High Capacity HEVC Video Hiding Algorithm Based on EMD Coded PU Partition Modes. *Symmetry* **2019**, *11*, 1015. [CrossRef]
48. Richardson, I.E. *H. 264 and MPEG-4 Video Compression: Video Coding for Next-Generation Multimedia*; John Wiley & Sons: Hoboken, NJ, USA, 2004.