applied sciences

MDPI

Article

Data Embedding in SHVC Video Using Threshold-Controlled Block Splitting

LieLin Pang [1,†], KokSheik Wong [2,†], Yiqi Tew [3,†] and Susanto Rahardja [4,*,†]

1 Faculty of Computer Science and Information Technology, University of Malaya, Kuala Lumpur 50603, Malaysia; adpangll@siswa.um.edu.my
2 School of Information Technology, Monash University Malaysia, Subang Jaya 47500, Malaysia; wong.koksheik@monash.edu
3 Faculty of Computing and Information Technology, Tunku Abdul Rahman University College, Kuala Lumpur 53300, Malaysia; yiqi@tarc.edu.my
4 School of Marine Science and Technology, Northwestern Polytechnical University, Xi'an 710072, China
* Correspondence: susantorahardja@ieee.org
† These authors contributed equally to this work.

Abstract: With the increasing number of video applications, it is essential to resolve issues such as ineffective search of video content, tampered/forged video content, packet loss, to name a few. Data embedding is typically utilized as one of the solutions to address the aforementioned issues. One of the important requirements of data embedding is to maximize embedding capacity with minimal bit rate overhead while ensuring imperceptibility of the inserted data. However, embedding capacity varies depending on the video content and increasing the embedding capacity usually leads to video quality degradation. In this work, a threshold-controlled block splitting technique is proposed for embedding data into SHVC video. Specifically, the embedding capacity can be increased by coding the host video by using more small blocks, which can be achieved by tuning a threshold-controlled parameter in the rate distortion optimization process. Subsequently, the predictive syntax elements in both intra and inter-coded blocks are jointly utilized to embed data, which ensures that data can be embedded regardless of the prediction mode used in coding a block. Results suggest that the proposed method can achieve a trade-off between the increase in embedding capacity and bit rate overhead while maintaining video quality. In the best case scenario, the sequence *PartyScene* can embed 516.9 kbps with an average bit rate overhead of +7.0% for the Low Delay P configuration, while the same video can embed 1578.6 kbps with an average bit rate overhead of +2.9% for the All Intra configuration.

Keywords: SHVC; data embedding; quad-tree block splitting; threshold-controlled

Citation: Pang, L.; Wong, K.; Tew, Y.; Rahardja, S. Data Embedding in SHVC Video Using Threshold-Controlled Block Splitting. *Appl. Sci.* **2021**, *11*, 4850. https://doi.org/10.3390/app11114850

Academic Editor: Juan J. Rodríguez

Received: 30 March 2021
Accepted: 15 May 2021
Published: 25 May 2021

1. Introduction

Nowadays, we rely on video technology for various purposes, including dashcam recording/surveillance video as evidence in the event of an incident, movie streaming for entertainment, and endoscopic video for health care. During the COVID-19 pandemic, many companies and organizations are also adopting video conferencing technology for virtual meeting to ensure business continuity. In addition, lecture/tutorial recording makes teaching and learning possible while teachers and students are physically apart.

As a result, several issues associated with video arise, including ineffective search of video content, unauthorized utilization, tampered/forged video content and packet loss. To address the aforementioned issues, data embedding, which inserts some data into the video, has gained increasing research interest [1]. For instance, hash codes are inserted into a video for efficient searching and retrieval purposes. Relevant information is embedded into a video to serve as hyperlinks for facilitating efficient exploration of large video repositories. Specifically, a hyper-linked video combines other videos by using a non-linear information structure, which allows a user to make choices based on the content of

the video as well as to navigate between videos and other hypermedia based on the user's interests [2]. In another application, error concealment suppresses interruption caused by abrupt bandwidth changes or packet losses by using data embedding techniques [3]. As one of the fastest growing technologies, machine learning requires huge a volume of data to train the machine learning model of interest. Ground truth, or more generally, metadata, can be inserted into a video as the training data to facilitate automated computer vision-based applications such as object tracking and localization. There are also security concerns over adversarial videos aiming to corrupt the training process [4], which can be addressed by data embedding to ensure that only genuine training videos are utilized. Furthermore, a biometric signature or fingerprint can be embedded into a video so that only authorized users can access the video content [5,6]. Moreover, a cryptographic hash of a video can be inserted into the video of interest so that the receiver can verify its originality [7].

Even with the recent breakthroughs in network and storage technologies, handling video in the raw (i.e., uncompressed) form is still a challenging task due to the large volume of data in the video. Therefore, most videos are stored and transmitted in the encoded form, each being compliant to specific video coding standards [8]. As an extension of the high efficiency video coding (HEVC) standard [9], the Scalable High Efficiency Video Coding (SHVC) standard [10] encodes a high definition video into a single bit stream with multiple resolutions/qualities. It emerges as an efficient video coding solution for adaptive video streaming applications. The scalability feature is essential, notably when considering the heterogeneity in the transmission environment and decoding devices. Many applications such as smart surveillance and monitoring systems have benefited from this technology. For example, it allows users at different locations to monitor an area in real-time at different display resolutions or frame rates depending on the network condition and constraint of the decoding device. However, the underlying architecture of SHVC differs from the previous scalable video coding standards such as H.264/SVC [11] or MPEG-2 [12]. As a result, the conventional data embedding methods cannot be applied directly in SHVC video. In addition, irregardless of the video standard in use, it is essential to design a data embedding method which can increase embedding capacity while maintaining the perceptual quality of the processed video. However, increasing the embedding capacity usually leads to greater distortion or quality degradation, and vice versa.

Therefore, in this work a data embedding method for SHVC video is proposed as illustrated in Figure 1. Specifically, a tunable threshold-controlled parameter is put forward to guide the rate distortion optimizer (RDO) to favor small split blocks over larger blocks. By tuning the threshold-controlled parameter to a higher value, more small split blocks are coded, which provides more predictive syntax elements that can be utilized for embedding data. Here, both intra and inter-coded blocks are jointly utilized to embed data into the smallest prediction blocks. Unlike the existing methods which are only able to embed data into blocks coded in either intra picture or inter picture prediction mode, this paper proposes a pair of data embedding techniques (viz., one for intra and another for inter picture prediction), where one of them can eventually be deployed to embed data regardless of the prediction mode used in coding the block. We increase the number of syntax elements, which in turns provides more venues for data embedding purposes. In addition, the proposed method is able to achieve a desired trade-off between the increase in embedding capacity versus bit rate overhead, which is crucial for applications such as error concealment, integrated video content management, video hyper-linking, steganography, and video annotation. For example, the efficiency of the video content search and retrieval system is particularly important for managing a huge volume of video files each of large size, which requires high embedding capacity. Therefore, this work aims to increase the embedding capacity without compromising the video quality. Our work makes the following contributions: (a) manipulating quad-tree block partitioning structure of a video frame so that more smaller blocks are coded; (b) minimizing the prediction error energy by selecting the best prediction mode for each block while matching the payload

bit to be embedded, and; (c) designing a pair of data embedding techniques, one for intra picture prediction and another for an inter picture prediction block, to ensure that data can be embedded regardless of the prediction mode used in coding a block.

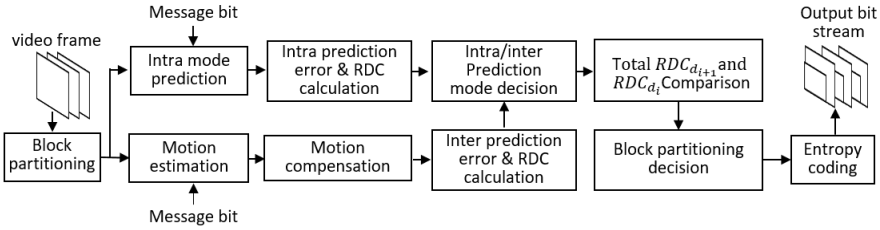


Figure 1. Flow of processes in the proposed method.

The rest of this paper is organized as follows: Section 2 briefly reviews the existing data embedding methods for compressed video. Section 3 describes the proposed data embedding technique in intra and intra-coded blocks. The experiment results are discussed in Section 4, and Section 5 concludes this paper.

2. Related Work

In this section, we briefly review methods designed for compressed video, where data embedding can take place during intra/inter picture prediction, transformation, quantization, and/or entropy coding.

In general, modifying an intra prediction directional mode to its adjacent directional mode has minimal impact on the video quality, although the file size is slightly increased. Therefore, this technique has been widely exploited to embed data in the literature. For example, Yang et al. [13] embed data by using the I4-blocks intra prediction mode in H.264/AVC video. They adopt matrix encoding to minimize the number of changes needed, viz., changing not more than one intra prediction mode when embedding two bits from the payload by using three I4-blocks. As newer standards, HEVC and its extension have more directional modes (i.e., more angles) for intra picture prediction, hence more data can be embedded by using the same technique. Sheng et al. manipulate the difference of two consecutive intra prediction directional modes or a pair of continuous planar and DC mode in HEVC to embed data [14]. To suppress the distortion caused by data embedding, Mareen et al. restrict the modification on the intra prediction mode by ± 1 so that the changes are imperceptible to the viewer [15]. Recently, Saberi et al. selected the smallest intra-coded prediction units in a coding tree unit (CTU) for data embedding purposes. In their work, a random function and an embedding threshold are defined to select CTUs for data embedding, but their proposed method suffers from low embedding capacity [16].

On the other hand, the motion vectors (MVs) of inter-coded blocks are also commonly exploited to embed data. For example, the least significant bit (LSB) of either the horizontal or vertical component of MV is modified to match one payload bit [17,18]. Instead of using all MVs, Aly only embeds data into MVs with high prediction errors in MPEG-2 video [19]. To increase the chances of successful data embedding, Van et al. exploit motion vector differences (MVD) and transformed coefficients in HEVC video [20]. Recently, Pang et al. proposed to associate payload bit with a motion vector predictor (MVP), which is utilized during inter picture prediction, to embed data without changing any MVs [21]. Changing the MVP causes insignificant/nearly no distortion to the video, although the difference between the MVP and the actual coded MV (denoted by MVD) is modified to embed data. While both the intra prediction mode and MVs are attractive techniques to embed data, there are situations where data cannot be embedded into a block. It is because the syntax element exploited for data embedding is simply unavailable; for example, the intra prediction mode is non-existence in an inter predicted block, hence the intra prediction mode cannot be exploited. Taking a different approach, Tew et al. manipulate the prediction block (PB) size and nonzero AC coefficients to increase the capacity [22], while Shanableh

associates each payload bit to a split flag which is utilized in HEVC to indicate whether a block is split into smaller blocks [23]. Yang et al. divide block partition modes into three groups to represent different data bits [24]. Although data are also embedded into the transformed coefficients [22,25–27], there are several drawbacks. Specifically, embedding data into the mid-frequency and high-frequency AC coefficients usually results in bit rate overhead, while embedding data in the low-frequency coefficients results in perceptual quality degradation. In addition, manipulating DC coefficients will change the entire block of pixel values. Therefore, embedding data into transformed coefficients may cause perceptible and notably degradation, which must be implemented appropriately in order to minimize the embedding artifact as well as error propagation.

To embed data into a scalable coded video, researchers often exploit the multi-layer coding structure defined in the scalable video coding standards. Specifically, a scalable coded video consists of a base layer (BL) and multiple enhancement layers (ELs). Shanableh exploits quantization parameters and MVs in all layers of an SNR scalable coded MPEG video to hide data [28]. Buhari et al. identify highly textured blocks in H.264/SVC video based on a predefined complexity model. They restrict data embedding to these identified blocks so that the error caused by coefficient manipulation is less noticeable [25]. Taking a similar approach, Amiri et al. restrict data embedding to regions with high texture and high contrast in H.264/SVC video [29]. Both Buhari et al. and Amiri et al. embed data by manipulating the AC coefficients. On the other hand, Pang et al. associate the data bit with the intra prediction mode in SHVC video when the selected mode disagrees with the payload bit [30]. The proposed method carries additional payload in multiple layers while suppressing video quality degradation by using a self-cancelling method, but it is only applicable to intra-coded blocks. Recently, Sun et al. embeded data into AC coefficients chosen by using a key-control selection algorithm in H.264/SVC [31].

While some data embedding methods have been put forward by researchers for scalable coded video, most methods are designed for an older generation of scalable video coding standard, namely H.264/SVC. This situation further motivated us to design a data embedding method for SHVC video.

3. Proposed Data Embedding Method

In this work, we propose a tunable threshold-controlled block splitting and data embedding method to increase the embedding capacity. Embedding capacity is the number of bits that can be embedded into the video sequence. Recall that, in SHVC, each video frame is partitioned into multiple blocks based on the quad-tree structure. Specifically, each frame is partitioned into multiple blocks called the *coding tree blocks* (CTBs) of size 64×64. Each CTB (i.e., at partitioning depth level 0, denoted by d_0) can in turn be split into multiple blocks called *coding blocks* (CBs) of size 32×32 (i.e., at level 1, denoted by d_1), where each CB can further be split into blocks of size 16×16 (i.e., at level 2, denoted by d_2), and eventually into blocks of size 8×8 (i.e., at level 3, denoted by d_3) as illustrated in Figure 2. Subsequently, a CB can be partitioned into multiple PBs. The smallest prediction block size for intra-coded block is 4×4 pixels, while the smallest prediction block size for inter-coded blocks is 4×8 or 8×4 pixels. During encoding, rate distortion optimization is performed to achieve a trade-off between the visual quality and the number of bits spent on coding as a rate distortion cost (RDC) function. The an encoder calculates $RDC = D + \lambda R$, where D is the distortion between the original signal and the reconstructed signals, while R is the compression rate which represents the total number of bits spent on coding. The parameter λ is the Lagrange multiplier used for Lagrangian optimization. Conceptually, coding a video frame using more (small) prediction blocks leads to more embedding opportunities due to the availability of the predictive syntax elements. Therefore, a threshold-controlled parameter τ is introduced to favor splitting of (small) blocks during rate-distortion optimization. The value of τ can be tuned to achieve a desired trade-off between the increase in capacity versus bit rate overhead while maintaining the perceptual quality of the output video. Specifically, when the scaled RDC

for a bigger block size (denoted by $(1+\tau) \times \mathcal{P}_d$) is larger than the accumulated *RDC* of the split blocks (denoted by $\mathcal{P}_{d+1}$), the split blocks option is adopted for coding instead of using the size suggested by RDO. Here,

$$\mathcal{P}_d = \sum_{i=0}^{n_d-1} RDC_d(i), \tag{1}$$

where n_d is the total number of split blocks at partitioning depth level d. An illustration of rate-distortion threshold guided split block is depicted in Figure 3. This assumes that x is the *RDC* at the current partitioning depth level d_i while y is the total *RDC* for the split block at depth level d_{i+1}. When $y < x \times (1+\tau)$, the split blocks option is adopted. It can be observed that more small blocks (highlighted in green) are coded when τ is set to some constant a. When τ is increased to some constant b where $b > a$, additional blocks are split into smaller blocks (highlighted in blue). These blocks offers additional venues for data embedding purposes. Note that the bit rate overhead can be adjusted by tuning the value of the threshold-controlled parameter.

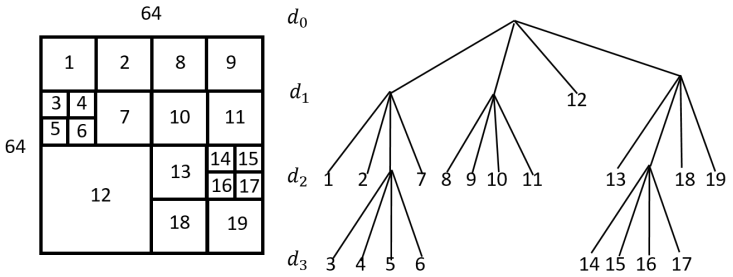


Figure 2. Example of block splitting.

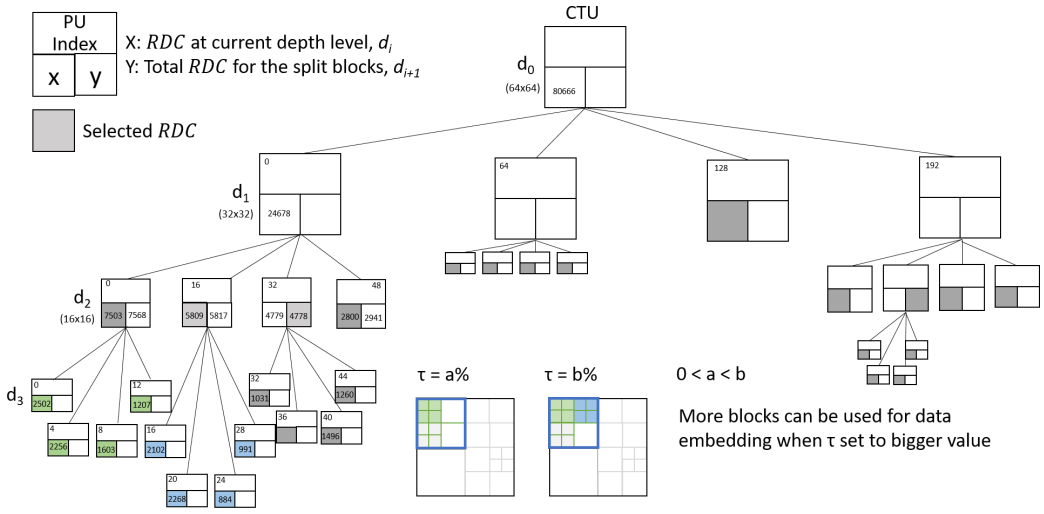


Figure 3. Example of threshold guided block splitting.

3.1. Data Embedding

To embed data, the payload bits are associated with the intra prediction modes of intra-coded blocks, as well as the MVP of the inter-coded blocks during intra and inter picture predictions, respectively. Let $M = \{m_i\}_1^N$ be the payload, where $m_i \in \{0,1\}$ and let N denote the length of M. The process in threshold guided block splitting and data embedding is shown in Algorithm 1. The RDC comparison process (i.e., line 19 to 22 in Algorithm 1) is performed for each partitioning depth level, and the process is repeated until all blocks are compared. Here, the intra prediction mode and MVP candidate selection decision in all scalable coded layers are manipulated to realize data embedding. Without

loss of generality, the descriptions for data embedding are based on a two-layer setting in a scalable coded video. Let ϑ denote the intra prediction mode, and let p denote the parity of the intra prediction mode, i.e., $p = f(\vartheta)$. When coding in intra mode, the RDO determines the best intra prediction mode to be coded. When the payload bit to be embedded differs from the parity of the best prediction mode (i.e., $m_i \neq p$) in the smallest PB, viz., 4×4, the intra prediction mode with the opposite parity bit with the best *RDC* from the candidate list is selected in place of the originally recommended one. In most cases, the second best prediction direction deviates slightly from the best intra prediction mode ϑ_{best}, viz., either $\vartheta_{best} + 1$ or $\vartheta_{best} - 1$. For example, when a payload bit m_i to be embedded in BL differs from the parity bit of the intra prediction mode p, the intra prediction direction angle is either increased or decreased to the next angle. The selection of optimum mode ϑ_{Best} is formulated below:

$$\vartheta_{\text{Best}} = \underset{p=m_i}{\arg\min}\{\text{RDC}(\vartheta, p)\}. \tag{2}$$

To facilitate further presentation, let the term *spatially co-located blocks* refer to two or more groups of blocks in different (resolutions) layers encoding the same visual information, but each in different resolution. Inter-layer prediction in SHVC uses the reconstructed video signal from a reference layer (denoted by EL_k, for $k < c$ where BL = EL_0) to improve the coding efficiency of the current enhancement layer EL_c. During inter-layer prediction, the co-located reconstructed blocks (viz., the visual information) from the reference layer EL_k are tagged with reference indices in the reference picture lists in addition to other prediction information in the current encoding layer EL_c. In order to achieve inter-layer prediction, EL requires the reconstructed pictures from the BL decoded picture buffer, which includes the reconstructed texture samples, and other prediction information [10]. Therefore, the intra prediction mode of the spatially co-located reconstructed block in the reference layer EL_k can be included as a candidate for the intra prediction mode in EL_i to improve the coding efficiency. When a block is selected for data embedding, the inter-layer intra prediction mode is set to the highest priority and labelled as the top performer in the candidate list so that it is selected for coding. Recall that the intra prediction mode assumes an integer value in the interval of $[0, 34]$, hence up to five bits (i.e., $\lfloor \log_2(35) \rfloor = 5$) can be embedded. Here, a threshold parameter ξ can be introduced to limit the number of bits to be embedded. To embed data, the sequence of bits induced by the intra prediction mode, namely $p_{\xi-1} p_{\xi-2} \cdots p_0$, is utilized to perform the bitwise-XOR operation with the payload bits m_i, i.e., $p'_i = p_i \oplus m_i$. The resulting sequence $\{p'_i\}$ is converted into a decimal value. The encoder will then compute the prediction error by comparing the predicted values (obtained by using this new mode) to the original ones. The prediction error is then transformed, quantized and entropy coded. Note that embedding more data using the intra prediction mode may lead to larger prediction error, and when the error is too large the other mode might be selected. Therefore, careful consideration is required when determining the number of bits embedded per intra predicted block.

Naturally, if a spatial scalable encoded video contains more layers, then more bits can be embedded. Likewise, when both intra- and inter-coded blocks are jointly utilized to embed data, more data can certainly be embedded. Therefore, we propose to utilize both intra prediction mode and MVP in all coded layers to embed data. Here, instead of using the MV of the inter-coded block, the parity bit of MVP indices is utilized to represent the payload bits. Recall that the predictive syntax element MVP was introduced in the HEVC and its extensions to improve the predictive coding of MVs for a coding block by using the neighboring motion information. Specifically, the difference between the MV of the block to be coded and the selected MVP is reduced (i.e., smaller range) so that the number of bits to be coded also becomes smaller. As an extension of the HEVC standard, the MVs from the spatial and temporally adjacent blocks are utilized as the MVP candidates.

Algorithm 1: Threshold-controlled Block Splitting and Data Embedding

```
TBS-DE(CU, d_i, M)
perform motion estimation
if (d_i == d_max) then
    obtain m_i from M
    associate m_i to MVP /* inter mode */
end
perform motion compensation
perform intra prediction
if (d_i == d_max) then
    embed m_i into IPM /* intra mode */
end
perform prediction error calculation
select intra or inter mode based on RDC
if (d_i < d_max) then
    split CU into cu_j, j ∈ 1..4
    for (j ← 1 to 4 ) do
        TBS-DE(cu_j, d_{i+1}, M)
    end
    calculate accumulated RDC, Pd_{i+1}
    if (P_{d_{i+1}} < (P_d × (1 + τ)) then
        P_d ← P_{d_{i+1}}
        CU ← cu_j, j ∈ 1..4
    end
end
return;
```

Then, an MVP can be flexibly selected by specifying the MVP index from this candidate list. Subsequently, MVD, which captures the difference between the current MV and the MVP, together with the MVP index, are encoded and signaled to the decoder. In particular, MVP is derived from the MVs of the spatially neighboring blocks (i.e., A_0, A_1, B_0, B_1 and B_2) and/or temporally co-located neighboring block (i.e., C_0, C_1, D_0 and D_1) as shown in Figure 4. The temporal MVP candidate for predicting MV for the current block can selected from more than one reference picture. For example, the first temporal MVP candidate is from the reference picture in the same coding layer while the second MVP candidate is from an inter-layer reference picture. Here, the selection of the MVP candidate depends on the value of $m_i \in \{0, 1\}$. Instead of using the MVP determined by RDO, motion information of the MVP candidate whose index (parity) matches m_i is selected for coding. Note that the selected MVP is either the best or first runner-up candidate with the minimum cost for inter picture prediction. The advantages of using MVP as a data carrier as compared to the conventional MV technique is that the predicted MV (magnitude and angle) remains unchanged (i.e., MV = MVP + MVD), hence the quality of inter-coded block can be preserved [21]. In addition, visual quality degradation in the processed video can be kept to the minimum by restricting data embedding to the smallest blocks (i.e., 4×4 in intra mode as well as 8×4 or 4×8 in inter mode) while skipping the rest. Therefore, in this work, only the smallest-coded blocks are leveraged to embed data. Beside that, a key κ can be utilized to select/skip blocks to distribute the payload bits across the entire video.

One of the advantages of joint utilization of the intra prediction mode and MVP for data embedding is that the payload is distributed into both intra and inter-coded blocks. The competition between the intra and inter prediction modes ensures that only the one with the better *RDC* is adopted for coding and data embedding purposes. Specifically, the RDO evaluates and decides the optimal mode (i.e., either intra or inter mode) to be coded. When the cost of embedding more bits in intra mode is higher than inter mode, inter mode will be coded, and vice versa. It is noteworthy that utilizing predictive syntax element to embed data can maintain the perceptual quality of the video without the introduction of drift error [32]. Specifically, the encoder calculates and encodes the prediction error/residual signal by taking the difference between the original and predicted

(after associated with data bits) values. Therefore, the reconstructed video is a close approximation of the original video.

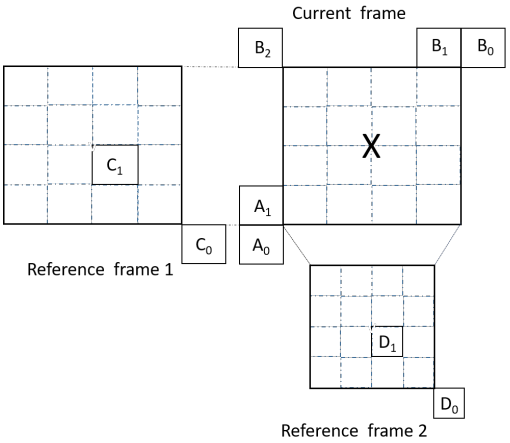


Figure 4. MVP candidates where 'X' is the current block to be coded.

3.2. Data Extraction

The embedded data can be extracted when decoding the processed video. Here, the same key κ, which is only known to the authorized parties, is utilized to identify the blocks selected for data embedding, i.e., which one is selected/skipped. The embedded data are then retrieved from the intra-coded block by extracting the *parity* of the intra prediction mode in BL, as well as the first ξ bits from the intra prediction mode in EL. Specifically, the embedded data in EL are derived from the coded mode and the spatial decoded mode of the spatially co-located block of the reference layer by using the bitwise-XOR operation. Similarly, the parity bits of the MVP indices are extracted from the inter-coded blocks in all coded layers.

The SHVC reference software SHM-12.0 [33] was modified to implement the proposed data embedding techniques. Here, we adopted the QP parameter settings as suggested in the SHM test conditions (CSTC) [34], i.e., QP = 22, 26, 30, and 34. Experiments were conducted by using two layers of spatial scalability for Low Delay P (LDP) and All Intra (AI) settings with a group of pictures (GOP) structure of 4, i.e., IPPPIPPP$\cdots$. The remaining parameters were set to the SHM default configuration. The experiments were conducted by using a PC with AMD Ryzen 5 3500U 2.10 GHz CPU and 8 GB of RAM running on a 64-bit Windows 10 operating system. Six standard video test sequences for SHVC from [35,36] as shown in Figure 5 were considered to evaluate the performance of the proposed data embedding method. A pseudo-random number generator was employed to generate a sequence of binary numbers (i.e., 0's and 1's), which was then embedded as the payload by using both the intra and inter picture prediction techniques. Unless specified otherwise, all available embedding venues were utilized, i.e., embedding at the maximum rate. Using SHVC compressed video as the baseline, the processed videos (containing embedded payload) were evaluated. It was verified that the resulting bit streams were still format compliant and they could be decoded by SHVC decoders while the embedded data could be extracted correctly for all settings. All results were collected by processing the first 200 frames in each video test sequence. It is crucial to note that the AI setting is considered here to capture the performance of embedding data by using the intra prediction mode, while the LDP setting captures the performance of embedding data by using both the intra prediction mode and the motion vector predictor.

(**a**) RushHour (**b**) FourPeople (**c**) BlueSky

(**d**) BasketballDrill (**e**) PartyScene (**f**) RaceHorses

Figure 5. Video test sequences considered for experiments in this work.

4. Experimental Results

4.1. Variation in Bit Rate

Let τ_0, τ_5 and τ_{10} refer to the setting when $\tau = 0$, 5 and 10%, respectively. Table 1 records the bit rate before and after data embedding by using the proposed techniques.

Table 1. Bit rate overhead (%) for the processed video.

Sequence	QP	LDP			AI		
		τ_0	τ_5	τ_{10}	τ_0	τ_5	τ_{10}
RushHour	22	0.4	6.5	21.0	0.3	3.3	8.1
@30 Hz	26	0.1	3.9	16.5	0.1	2.6	6.8
BL(960 × 540)	30	0.1	3.7	13.4	0.0	2.1	5.9
EL(1280 × 720)	34	0.0	4.6	13.0	0.0	2.1	5.7
	Average	0.1	4.3	15.2	0.1	2.4	6.3
FourPeople	22	1.3	8.9	16.3	1.2	3.7	6.7
@60 Hz	26	1.5	7.3	13.9	1.2	3.6	6.5
BL(640 × 360)	30	1.5	6.4	12.1	1.0	3.3	6.5
EL(1280 × 720)	34	0.3	6.0	13.3	0.5	3.1	6.6
	Average	1.2	6.8	13.3	1.0	3.5	6.6
BlueSky	22	0.5	4.6	8.0	0.6	3.3	6.2
@24 Hz	26	0.7	5.5	10.1	0.6	3.5	6.6
BL(640 × 360)	30	0.8	6.0	11.4	0.4	3.8	7.5
EL(1280 × 720)	34	0.5	7.2	15.2	0.0	4.3	9.0
	Average	0.7	6.0	11.6	0.4	3.8	7.6
BasketballDrill	22	0.8	7.0	13.2	1.1	3.9	7.1
@50 Hz	26	0.9	7.4	15.1	1.0	3.7	7.0
BL(416 × 240)	30	0.9	7.9	16.4	1.0	3.7	7.2
EL(832 × 480)	34	0.6	8.9	18.1	0.8	3.5	7.5
	Average	0.8	7.8	15.9	0.9	3.6	7.1
PartyScene	22	0.7	4.3	7.0	0.9	4.1	2.9
@50 Hz	26	1.1	5.7	9.7	1.1	3.4	4.9
BL(416 × 240)	30	1.2	7.2	13.4	1.2	3.9	5.9
EL(832 × 480)	34	0.6	9.9	19.1	0.9	4.6	7.6
	Average	1.0	7.2	13.3	1.0	4.0	6.0
RaceHorses	22	0.4	4.3	8.7	0.2	5.0	7.8
@30 Hz	26	0.5	5.3	11.1	0.3	4.9	8.3
BL(416 × 240)	30	0.5	6.5	14.9	0.3	5.2	9.5
EL(832 × 480)	34	0.4	9.8	20.7	0.0	5.9	11.6
	Average	0.5	6.8	14.8	0.3	5.2	9.6

Results suggest that, for both LDP and AI configurations, the bit rate increases as τ increases. This is an expected result because larger τ permits more split blocks to be coded (hence increasing embedding capacity) instead of larger blocks which are more cost effective. The lowest average bit variation is observed in *BlueSky* for the LDP configuration and *PartyScene* for the AI configuration. A reason for such outcome is that these sequences are originally coded with 80∼90% of small blocks at partitioning depth level d_3 in both layers. On the other hand, the largest average bit variation is observed in *BasketballDrill* for the LDP configuration and *RaceHorses* for the AI configuration. A reason for such outcome is due to large homogeneous regions in both video sequences. These smooth regions can be effectively coded by using larger blocks, but instead they are partitioned into smaller blocks for increasing the embedding capacity. Based on these observations, we conclude that: (a) a video sequence with textured scene offers higher embedding capacity and; (b) the total number of the smallest sized PB to be coded can be controlled by using τ so that barely sufficient capacity is offered to accommodate the data, without over-supplying, which leads to a higher bit rate overhead.

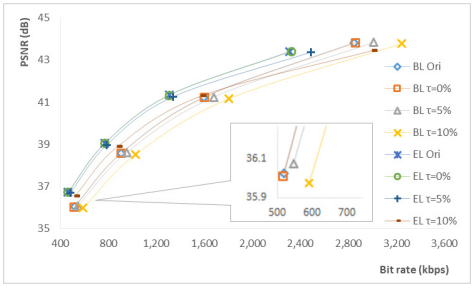
4.2. Video Quality

The effect of data embedding on video quality is reported in Table 2. In all cases, the average PSNR degradation is less than 0.17 dB and 0.13 dB for the LDP and AI configurations, respectively. In terms of SSIM [37], the degradation is less than 0.0027 and 0.0022 for the LDP and AI configurations, respectively. The sequence *PartyScene* experiences slightly greater degradation due to its high embedding capacity for all considered τ values. On the other hand, the lowest average quality degradation is observed in the sequence *BlueSky* and *RushHour* for the AI configurations, respectively. It is also observed that video coded by using the AI configuration is slightly inferior in terms of video quality. It is mainly caused by the significantly higher payload carried by the all-intra-coded sequences. Nonetheless, the results for both configuration settings show comparable video quality after embedding data, and these outcomes suggest that the manipulation of PB size has negligible impact on the video quality for all video sequences.

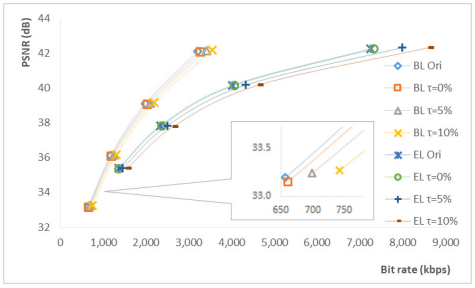
To visualize the performance of the proposed technique, we also plot the RD curves comparing the original (compressed) and processed videos for the LDP and AI configurations in Figures 6 and 7, respectively. Results suggest that data embedding leads to a slight drop in coding efficiency. The graphs also suggest that implementing the proposed techniques at higher QP causes greater loss in coding efficiency, and vice versa. To further analyze the video quality degradation and bit rate overhead, the Bjøntegaard Delta of bit rate (BD-RATE) and Bjøntegaard Delta of PSNR (BD-PSNR) are computed [38]. The average bit rate overhead and visual quality degradation caused by data embedding are recorded in Table 3. The results suggest that the average bit rate overheads for the LDP and AI configurations fall in the ranges of $[1.01, 13.84]$ and $[1.51, 7.07]$ percent, respectively. On the other hand, the average quality degradation for LDP and AI configurations fall in the ranges of $[0.05, 0.63]$ dB and $[0.10, 0.45]$ dB, respectively.

Table 2. Visual quality degradation in term of PSNR (dB) and SSIM for the processed video.

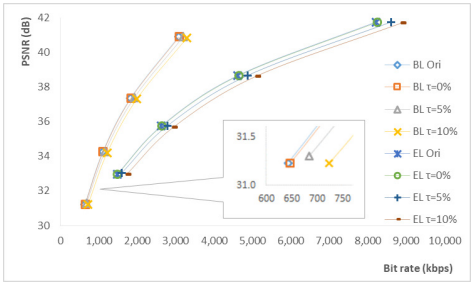
Sequence	QP		PSNR (10^{-2})						SSIM (10^{-4})					
			LDP			AI			LDP			AI		
			τ_0	τ_5	τ_{10}	τ_0	τ_5	τ_{10}	τ_0	τ_5	τ_{10}	τ_0	τ_5	τ_{10}
RushHour	22	BL	0.6	0.0	5.4	3.5	0.0	1.3	0.2	0.0	0.3	0.5	0.0	0.0
		EL	0.0	3.2	0.0	1.1	0.0	0.0	0.0	1.0	0.0	0.1	0.0	0.0
	26	BL	0.6	0.0	7.1	3.6	0.0	0.0	0.0	0.0	1.2	0.9	0.0	0.0
		EL	0.8	8.0	2.8	1.3	0.0	0.0	0.2	3.4	2.1	0.1	0.0	0.0
	30	BL	0.7	0.0	7.9	2.7	0.0	0.0	0.2	0.0	3.1	0.7	0.0	0.0
		EL	0.0	6.9	13.1	1.1	0.0	0.0	0.1	4.3	10.6	0.2	0.0	0.0
	34	BL	1.6	0.0	5.3	1.8	0.0	0.0	2.1	0.0	0.9	0.0	0.0	0.0
		EL	0.2	2.2	16.4	0.7	0.0	0.0	1.0	2.7	19.7	0.0	0.0	0.0
FourPeople	22	BL	3.9	0.0	0.0	10.3	6.4	8.1	1.4	0.0	0.0	3.0	1.3	1.4
		EL	0.2	0.0	0.0	2.5	0.3	2.5	0.0	0.0	0.0	0.2	0.0	0.0
	26	BL	2.6	0.0	0.0	9.9	6.3	8.6	1.6	0.0	0.0	5.3	2.6	3.0
		EL	0.6	0.0	0.0	3.3	0.3	2.8	0.1	0.3	0.1	0.3	0.0	0.0
	30	BL	4.1	0.0	0.0	10.4	4.5	4.8	3.0	0.0	0.0	9.9	2.1	0.1
		EL	1.4	0.3	3.9	3.5	0.0	1.6	0.0	1.0	2.2	0.4	0.0	0.0
	34	BL	4.4	0.0	0.0	9.2	2.1	1.6	6.4	0.0	0.0	13.2	0.0	0.0
		EL	2.5	0.8	1.1	3.8	0.0	0.0	0.3	1.6	2.2	0.9	0.0	0.0
BlueSky	22	BL	1.0	0.5	5.9	6.3	0.0	1.1	0.5	0.0	0.0	2.7	0.4	0.1
		EL	0.2	0.0	4.6	2.8	0.0	0.0	0.1	0.0	0.0	0.1	0.0	0.0
	26	BL	0.4	0.0	4.0	5.2	1.8	8.4	0.4	0.0	0.0	4.1	0.4	0.9
		EL	0.0	0.0	2.8	3.2	0.0	3.1	0.0	0.0	0.0	0.4	0.0	0.0
	30	BL	0.4	0.0	5.7	5.2	3.3	12.0	0.6	1.5	2.3	6.6	1.1	2.2
		EL	0.0	0.0	6.7	3.6	0.0	3.4	0.0	0.0	0.0	0.8	0.0	0.0
	34	BL	0.4	0.0	0.2	5.7	2.1	9.5	0.3	0.0	1.7	10.0	0.6	2.2
		EL	0.2	0.0	0.0	4.3	0.0	0.0	0.0	0.0	0.0	1.2	0.0	0.0
BasketballDrill	22	BL	2.9	0.0	3.5	7.4	5.6	7.7	2.5	0.3	4.1	4.4	4.1	5.9
		EL	1.1	0.4	5.6	4.1	1.4	1.4	0.0	0.0	1.1	1.4	0.0	0.0
	26	BL	3.0	1.4	3.9	6.3	4.7	6.6	3.5	1.7	5.7	6.1	6.0	6.2
		EL	0.4	0.9	4.4	4.4	1.9	1.7	0.0	0.4	3.3	1.9	1.1	0.3
	30	BL	0.4	0.0	1.8	7.1	4.2	6.2	2.9	0.0	0.0	8.6	5.7	7.2
		EL	0.9	0.9	3.0	3.9	0.5	1.1	1.5	0.9	0.0	2.3	0.0	0.9
	34	BL	1.5	0.0	0.0	7.0	3.4	3.3	3.1	0.0	0.0	9.9	2.6	0.0
		EL	0.1	0.0	1.7	3.8	0.0	0.0	0.0	0.0	0.0	1.9	0.0	0.0
PartyScene	22	BL	1.9	0.0	4.0	13.0	7.1	6.2	0.6	0.0	2.8	4.8	2.1	2.6
		EL	0.7	0.0	2.0	6.4	0.0	0.0	0.0	0.0	0.4	1.0	0.0	0.0
	26	BL	1.3	0.1	4.1	12.2	5.3	6.2	1.2	1.3	7.5	8.3	3.4	4.0
		EL	0.0	0.0	1.4	5.5	0.0	0.0	0.0	0.0	1.2	1.8	0.0	0.0
	30	BL	1.5	0.6	4.6	10.7	3.2	3.7	2.0	7.3	20.6	15.0	4.1	4.0
		EL	0.5	0.0	1.0	5.4	0.0	0.0	0.0	0.0	1.3	2.7	0.0	0.0
	34	BL	3.8	0.0	4.1	9.9	1.2	0.7	11.3	8.7	27.0	21.6	5.4	4.6
		EL	0.9	0.0	0.0	6.1	0.0	0.0	0.0	0.0	0.0	5.2	0.0	0.0
RaceHorses	22	BL	1.3	0.3	6.1	7.4	0.0	1.8	1.3	0.0	2.5	3.8	0.3	0.0
		EL	0.1	2.6	5.5	4.2	0.0	0.0	0.0	0.7	0.6	0.3	0.0	0.0
	26	BL	1.4	1.1	6.0	6.7	1.0	3.7	8.2	3.9	10.0	4.7	0.9	2.4
		EL	0.0	1.4	5.7	4.1	0.0	0.0	0.6	0.4	2.7	0.9	0.0	0.0
	30	BL	1.5	0.0	0.7	6.0	0.7	0.6	0.0	0.0	0.0	7.4	0.2	0.0
		EL	0.3	1.0	4.0	4.1	0.0	0.0	0.0	6.8	8.0	1.7	0.0	0.0
	34	BL	1.1	0.0	0.0	5.6	0.0	0.0	0.0	1.6	6.1	16.1	4.5	0.0
		EL	0.2	0.0	0.0	4.7	0.0	0.0	0.0	0.0	0.0	5.2	0.0	0.0



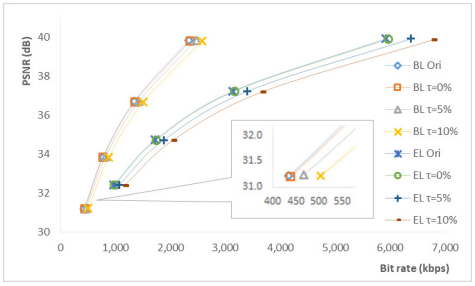
(**a**) RushHour



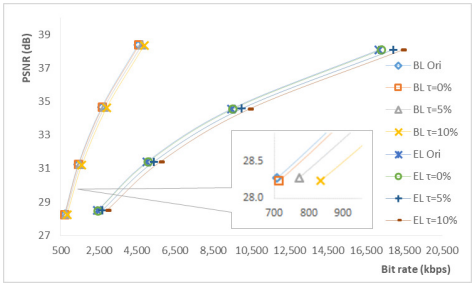
(**b**) FourPeople



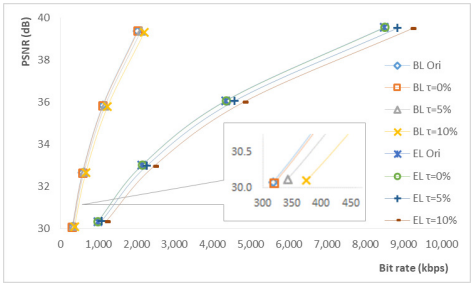
(**c**) BlueSky



(**d**) BasketballDrill



(**e**) PartyScene

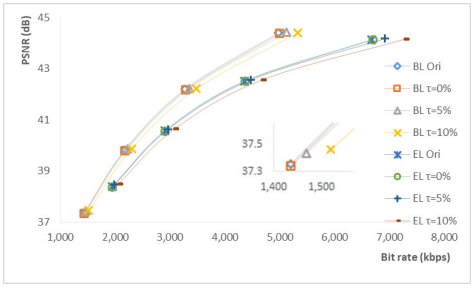


(**f**) RaceHorses

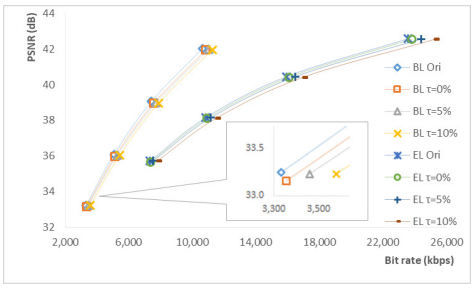
Figure 6. Rate distortion curve for the original compressed videos (Ori) and the processed videos using LDP configuration.

Table 3. Comparison in term of BD-RATE and BD-PSNR for the processed video.

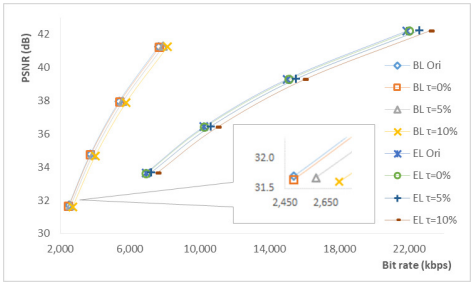
Sequence	BD-BR (%)						BD-PSNR (dB)					
	LDP			AI			LDP			AI		
	τ_0	τ_5	τ_{10}	τ_0	τ_5	τ_{10}	τ_0	τ_5	τ_{10}	τ_0	τ_5	τ_{10}
RushHour	0.26	4.76	17.63	0.44	1.42	5.35	0.01	0.20	0.70	0.02	0.07	0.27
FourPeople	1.91	6.23	12.30	2.11	3.86	7.13	0.10	0.29	0.55	0.14	0.26	0.47
BlueSky	0.71	5.17	11.36	1.01	3.73	7.99	0.04	0.28	0.61	0.08	0.30	0.63
BasketballDrill	1.13	7.42	15.83	2.02	4.10	7.58	0.05	0.33	0.68	0.11	0.22	0.40
PartyScene	1.34	6.23	12.04	2.17	3.87	5.84	0.07	0.32	0.60	0.16	0.29	0.43
RaceHorses	0.73	5.99	13.89	1.31	4.39	8.53	0.03	0.27	0.61	0.08	0.26	0.49
Average	1.01	5.97	13.84	1.51	3.56	7.07	0.05	0.28	0.63	0.10	0.23	0.45



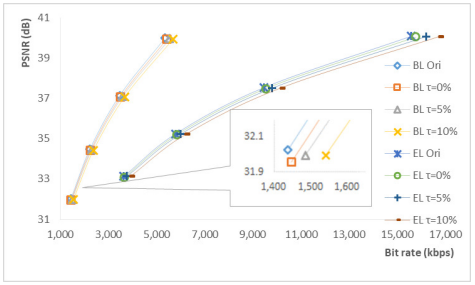
(**a**) RushHour



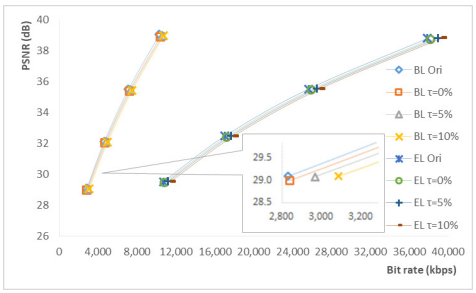
(**b**) FourPeople



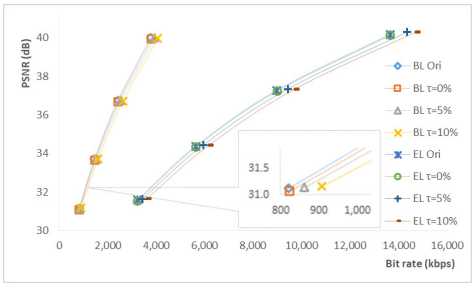
(**c**) BlueSky



(**d**) BasketballDrill



(**e**) PartyScene



(**f**) RaceHorses

Figure 7. Rate distortion curve for the original compressed videos and the processed videos using the AI configuration.

4.3. Embedding Capacity

The embedding capacity of each video using the proposed method is recorded in Tables 4 and 5 for the LDP and AI configurations, respectively. Generally, video sequences with complex texture or fast-moving scenes (e.g., *PartyScene* and *BasketballDrill*), which are predicted by using more small PBs, can offer more rooms for data embedding, and vice versa (e.g., *RushHour*). In the best case scenario, the sequence *PartyScene* embeds 516.9 kbps with an average 7.0% bit rate overhead for the LDP configuration, while it embeds 1578.6 kbps with an average 2.9% bit rate overhead for the AI configuration. On the other hand, the sequence *RushHour*, which contains smooth regions of different sizes and mostly static background (due to the air ripples caused by heat) offers the smallest embedding capacity. The embedding capacity for this video sequence is expected to be small because most slices in the video are coded by using large PB.

Table 4. Embedding capacity (kbps) for the proposed embedding technique using LDP configuration.

Sequence	QP	τ_0			τ_5			τ_{10}		
		BL	EL	Total	BL	EL	Total	BL	EL	Total
RushHour @30 Hz	22	15.7	6.7	22.4	42.9	36.1	79.1	79.5	138.8	218.3
	26	6.4	2.3	8.7	19.0	12.0	31.1	35.2	45.8	81.0
	30	2.3	0.7	3.0	7.7	4.4	12.2	15.7	17.5	33.2
	34	0.7	0.2	0.8	2.6	1.3	3.9	6.1	6.5	12.6
FourPeople @60 Hz	22	43.6	79.7	123.3	78.7	182.4	261.1	106.6	296.7	403.2
	26	28.7	49.2	77.9	54.0	107.0	161.0	78.3	178.1	256.4
	30	14.9	24.7	39.6	31.0	56.6	87.6	48.2	99.8	148.0
	34	5.0	8.1	13.1	12.1	20.7	32.8	22.1	43.4	65.6
BlueSky @24 Hz	22	21.5	78.9	100.4	47.5	174.1	221.6	66.3	242.4	308.7
	26	14.1	47.1	61.2	35.5	120.4	155.9	54.4	185.0	239.3
	30	7.5	22.3	29.8	25.4	78.7	104.0	41.1	126.8	167.9
	34	3.5	7.9	11.5	13.3	35.1	48.3	29.6	82.9	112.6
BasketballDrill @50 Hz	22	23.0	67.6	90.6	44.8	141.8	186.6	65.6	218.5	284.1
	26	14.8	34.9	49.7	28.7	72.2	100.9	44.1	113.0	157.1
	30	8.4	18.8	27.2	17.4	38.8	56.1	27.9	64.6	92.5
	34	3.5	7.9	11.4	8.2	18.6	26.9	14.9	33.5	48.4
PartyScene @50 Hz	22	43.9	190.5	234.4	86.7	341.2	427.9	108.6	408.4	516.9
	26	30.7	142.1	172.8	63.7	255.4	319.1	86.8	316.1	402.9
	30	17.4	91.2	108.6	39.5	172.7	212.2	59.5	225.5	285.0
	34	6.6	37.8	44.4	17.9	85.2	103.1	31.4	127.9	159.4
RaceHorses @30 Hz	22	17.7	44.6	62.3	36.1	128.9	165.0	54.9	184.8	239.6
	26	12.2	32.1	44.3	24.2	88.0	112.2	38.7	131.7	170.4
	30	6.9	18.6	25.5	13.9	51.0	64.9	22.5	81.8	104.3
	34	3.2	8.3	11.4	6.9	21.5	28.5	12.0	35.6	47.6

Table 5. Embedding capacity (kbps) for the proposed embedding technique using AI configuration.

Sequence	QP	τ_0			τ_5			τ_{10}		
		BL	EL	Total	BL	EL	Total	BL	EL	Total
RushHour @30 Hz	22	28.3	41.9	70.1	81.7	139.3	221.0	163.4	311.5	474.8
	26	15.7	21.2	37.0	44.5	69.3	113.8	96.0	163.2	259.1
	30	7.1	9.3	16.3	21.2	30.4	51.6	48.8	77.4	126.2
	34	2.6	3.2	5.8	8.7	11.7	20.4	22.0	32.8	54.8
FourPeople @60 Hz	22	175.8	401.4	577.3	290.9	828.9	1119.8	376.1	1251.0	1627.1
	26	138.7	304.7	443.4	235.5	602.1	837.7	315.5	921.0	1236.5
	30	95.5	202.2	297.7	171.9	406.0	577.9	245.4	642.0	887.4
	34	52.9	112.4	165.3	106.4	238.8	345.1	165.7	409.4	575.1
BlueSky @24 Hz	22	57.2	301.8	359.0	126.3	607.7	734.0	181.3	835.0	1016.3
	26	44.5	228.8	273.3	107.1	500.6	607.7	161.6	722.3	883.9
	30	30.5	153.8	184.3	87.2	388.5	475.8	141.9	608.9	750.8
	34	18.0	86.0	104.0	65.1	273.5	338.7	119.0	488.1	607.1
BasketballDrill @50 Hz	22	67.3	353.8	421.1	131.3	657.6	788.9	186.9	938.4	1125.3
	26	47.6	198.8	246.4	89.7	374.7	464.4	131.3	570.7	701.9
	30	31.7	112.7	144.4	60.8	207.4	268.2	90.4	316.6	406.9
	34	18.9	65.0	83.9	38.8	122.3	161.1	62.5	187.9	250.4
PartyScene @50 Hz	22	135.3	752.2	887.5	233.5	1125.0	1358.5	271.6	1307.1	1578.6
	26	114.0	625.1	739.2	200.8	939.6	1140.3	245.8	1120.3	1366.2
	30	83.0	498.7	581.7	157.0	756.3	913.2	206.3	919.6	1125.9
	34	47.8	345.0	392.8	105.5	567.4	672.9	155.2	724.1	879.3
RaceHorses @30 Hz	22	32.9	134.5	167.4	75.3	383.0	458.3	116.2	525.7	642.0
	26	26.9	109.3	136.2	58.9	297.9	356.8	94.9	429.0	523.9
	30	19.0	83.6	102.6	42.0	221.9	263.9	71.6	337.7	409.3
	34	11.2	50.2	61.4	23.1	143.9	167.0	41.3	236.6	277.8

Results also suggest that when τ is set to a larger value, higher embedding capacity is achieved. This trend is most apparent for the sequence *RushHour* because, when τ increases, more split blocks are coded instead of the block sizes suggested by RDO. Hence, a trade-off between embedding capacity and bit rate overhead can be achieved by tuning τ. It is observed that AI achieves, in general, a higher embedding capacity in comparison to LDP. The largest difference is observed in the sequence *FourPeople*, followed by *BasketballDrill*, *BlueSky*, *PartyScene*, *RaceHorses* and *RushHour*. Notably, for the sequence *FourPeople*, AI achieves, on average, $\sim$5.9 $\times$ more embedding capacity for τ_{10} in comparison to LDP. This is because the sequence *FourPeople* contains 70% static background scenes. On the other hand, for the sequence *RushHour*, AI achieves 2.2$\sim$4.3 $\times$ more embedding capacity in comparison to LDP for τ_{10}.

4.4. Comparison with Conventional Methods

For the purposes of fair comparison and analysis, the experiment environment and data set should be exactly the same, viz., using the same video encoder settings and the same video test sequences. However, we face three major challenges: (a) different video test sequences are used in the literature; (b) most experiments reported in the literature are conducted by using earlier video coding standards, and; (c) incomplete information about the parameter settings used for conducting experiments. These challenges prevent us from conducting and furnishing a thorough, fair and meaningful experiment using the conventional methods. Despite these challenges, we do however still provide a comparative analysis based on the available information for the completion of discussion.

Specifically, for performance comparison, we define the embedding cost $\gamma = (n_e - n_o)/\rho$ as the number of bits spent on the video test sequence for encoding one payload bit, where n_o and n_e are the numbers of bits spent on coding (bit stream size) the original and processed video sequences, respectively, and ρ is number of payload bits embedded into the video sequence of interest. Based on the results, for τ_0, on average, 1.65 and 1.40 bits are required to embed 1 bit for the LDP and AI configurations, respectively.

In comparison, to embed one payload bit into the video, Aly et al's method for MPEG-2 video requires 5.36 bits, while Noorkami et al.'s method for H.264 video requires 1.50 bits [39]. On the other hand, Mareen et al.'s embedding method implemented by using HEVC requires 1.69 bits [15]. When τ is increased, more bits are spent to embed each payload bit because split blocks might be coded (instead of a bigger block), which required some signaling to the decoder. In other words, the increase in bit stream size is due to data embedding (i.e., a sub-optimal mode is selected hence larger prediction error) as well as additional signaling to the decoder. For example, for τ_5, the embedding cost falls in the range of $[3.09, 6.75]$ and $[1.96, 3.28]$ for the LDP and AI configurations, respectively. Similarly, for τ_{10}, the ranges are $[4.06, 6.84]$ and $[2.30, 3.43]$ for the LDP and AI configurations, respectively.

Tables 6 and 7 record the embedding capacity for the proposed and conventional methods [15,20,21,30] using the same QP settings. It is observed that the proposed method (which uses multiple syntax elements) achieves higher embedding capacity in comparison to the conventional methods (which uses a single syntax element) considered. The embedding capacity is doubled for LDP when both MVP and intra prediction mode are utilized for data embedding. It is also observed that higher embedding capacity is achieved for AI configuration. This is because more prediction modes are manipulated for data embedding in the proposed method.

A relative functional comparison is conducted by considering the conventional methods proposed for MPEG-2 [19], H.264/AVC [39,40], H.264/SVC [25,31], and HEVC [15,16,20,22–24,41]. Table 8 compares the aforementioned methods in terms of embedding venue, applicability to intra/inter picture prediction, embedding capacity, video quality and bit rate overhead. Here, a relative comparison is performed because the results and values reported by the respective authors in the existing works are collected by using different test videos, video coding standards, parameter settings, etc. The labels of high (H), moderate (M)

and low (L) are context-dependent and they are relative in nature. Firstly, for embedding capacity, we assign one of the three labels depending on the number of available syntax elements in a video. Specifically, more syntax elements provide more venues for data embedding, and vice versa [42]. While classical venues such as the coefficient offer high embedding capacity, recently identified venues (e.g., MVD in HEVC) will, in general, offer less embedding capacity. Nonetheless, the proposed technique has the flexibility to embed different amounts of payload by adjusting τ. When the required embedding capacity is low, τ_0 can be adopted so that the bit rate overhead and distortion can be reduced. On the other hand, more embedding capacity could be achieved by using a larger τ so that each CTU can be split into smaller blocks. Note that the total number of smaller blocks in a CTU is determined by the RDC, and it is guided by τ so that block-splitting will not be performed when the cost is higher than τ.

Table 6. Comparison of embedding capacity (kbps) for the LDP configuration.

Sequence	τ_0			τ_5		τ_{10}	
	[20]	[21]	Proposed	[21]	Proposed	[21]	Proposed
RushHour	11.44	12.1	22.04	46.84	81.16	145.18	222.96
FourPeople	13.12	14.71	80.32	39.58	179.96	59.3	297.27
BlueSky	28.63	33.88	69.18	65.01	161.98	75.34	261.31
BasketballDrill	19.88	22.48	71.83	41.04	156.86	54.18	245.52
PartyScene	47.76	56.53	144.87	102.78	332.51	118.92	447.22

Table 7. Comparison of embedding capacity (kbps) for the AI configuration.

Sequence	[30]	[15]	Proposed
RushHour	48.58	52.22	57.81
FourPeople	180.89	285.96	346.73
BlueSky	114.52	116.67	206.29
BasketballDrill	141.71	171.75	281.63
PartyScene	238.05	304.11	462.46

Table 8. Relative functional comparison of conventional and proposed data embedding techniques within various video coding standards.

Functional Comparison	Embedding Venue	Applicable to		Embedding Capacity	Quality after Embedding	Bit Rate Overhead
		I-Frame	P/B Frame			
[15,16,30,40]	Intra prediction mode	✓	✓	H	H	M
[19,20]	MVD	X	✓	L	H	L
[23,24]	Block partition	✓	✓	H	M	M
[25,31,39,41]	Coefficient	✓	✓	H	M	M
[43]	Merge mode	X	✓	M	H	L
[22]	Block partition + Coefficient	✓	✓	H	M	M
Proposed	Intra prediction mode + MVP	✓	✓	H	H	M

H: High, M: Moderate, and L: Low.

Next, we analyze the impact of data embedding on the quality of the video. It is observed that the quality of the video may be affected depending on when (i.e., at which stage) data are embedded within the video encoding process as well as the embedding capacity. For example, when data embedding takes place before the computation of the prediction error/residual such as using intra prediction mode and MV, the residual will be obtained by using the new predicted value; therefore, the quality of the video can be maintained, although the residual is slightly larger. On the other hand, manipulating coding block structure may affect the selection of the optimal prediction block size and prediction mode, which will indirectly cause higher prediction error. Therefore the quality of the video is affected. Similarly, manipulating the transformed coefficients causes the

entire block of pixel values to change. The perceived quality of these two methods will then be affected by the quantization process applied on the prediction residues. Similar findings are observed from related works; for example, the perceptual quality degradation for the intra prediction mode and merge mode is less than 0.2 dB [15,16,40] while MVD [19] is less than 0.06 dB. Since the quality degradation is less than 1 dB, which is negligible, therefore we label the quality as 'H'. In addition, it is observed that the quality degradation for coefficient-based methods [25,39] and coding block structure-based methods [22–24] falls in the range of $[0.2, 4.8]$ dB and $[1.0, 2.3]$ dB, respectively. Since the quality is slightly inferior in comparison to the former, it is labeled'M' for these methods.

Subsequently, we analyze the impact of data embedding on the bit rate based on these works. Generally, when more bits are embedded into a video sequence, the bit rate overhead will be higher. Besides that, more bits are required to code the non-optimal block structure or predicted values, as well as additional syntax elements resulting from data embedding. Therefore, when more changes to the syntax elements or more syntax elements are coded due to data embedding, the bit rate overhead increases accordingly. These findings are consistent with the observation. Specifically, it is observed that the number of MVD is relatively low as compared to other syntax elements, and the bit rate overhead incurred is less than 1% [19]. A coding block and its neighboring blocks are usually correlated with each other and they contain the same moving object with similar motion. The MV after mapping the merging block to the payload bit is likely to be maintained. Therefore, the bit rate overhead after manipulating merge mode can be maintained at a relatively low value (e.g., 2.3% as reported in [43]). Since the bit rate overhead is maintained at a minimum level, these methods are labeled as 'L'. On the other hand, the bit rate overhead caused by using the intra prediction mode to embed data in [15] is capped at 4%, while the bit rate overhead caused by manipulating coding block structure falls in the range of [5, 5.5] in Yang et al.'s method [24]. In addition, the bit rate overhead caused by manipulating coefficient falls in the range of [1.4, 5.0] in Noorkami et al.'s method [39] and Buhari et al.'s method [25]. Since the bit rate overhead for these methods is relatively higher in comparison to the former methods, they are labeled as 'M'.

In general, all considered methods embed data during the encoding process. However, depending on the eventual mode of encoding of a block (i.e., MB in H.264 or CU in HEVC/SHVC), most existing methods may or may not be able to embed data. For example, the intra picture prediction-based method will not be able to embed data into a block when the inter picture prediction mode turns out to be more cost effective (hence the block is coded by the using inter prediction mode), and vice versa. On the other hand, the proposed method is able to embed data into both intra and inter predicted blocks. In other words, eventually one mode will be chosen (i.e., intra or inter), and in any case, data can be embedded by the proposed method. Similar to the proposed method, Tew et al.'s method [22] can embed data in each CU since block size is exploited, while Shanableh [23] offers the same capability by exploiting the block-splitting flag.

Since technology constantly advances to newer standards, we also evaluate the feasibility of applying the conventional data embedding methods to the SHVC standard. The MVD in [19] is not feasible for data embedding in EL when an inter-layer reference picture is utilized in EL, because all MVs are set to *zero* to fulfil the requirement of bit stream conformance in EL(s) [44]. On the other hand, when the fast encoder mode setting is enabled, only blocks with size $2N \times 2N$ are coded, hence any method that is based on block partitioning (e.g., [23]) can only be applied to BL, unless the fast encoder mode is disabled. Likewise, although a coefficient-based method can be easily adopted in SHVC, propagation of error should be handled carefully to avoid poor perceptual quality for inter-layer predicted picture.

All in all, in comparison to AVC-based and HEVC-based methods, the achievable embedding capacity in scalable coded video by the proposed method is encouraging because the syntax elements in all coded layers can be exploited for data embedding purposes. Furthermore, some existing techniques can be combined with our work to achieve better

trade-off among embedding capacity, quality, and bit rate. For example, the PB embedding technique can be applied to BL while the MVD, coefficient and merge mode can be applied to all layers to achieve a higher combined embedding capacity. This trade-off will be further investigated as part of our future work.

5. Conclusions

In this paper, a tunable threshold-controlled block splitting and data embedding method is proposed. The embedding capacity is improved by making the PBs assume smaller sizes, which directly results in having more manipulable predictive syntax elements for data embedding purposes. Specifically, the strength of our proposed method is that the embedding capacity can be tuned according to the varying payload requirements. When the required embedding capacity is low, the room given to code more smaller blocks can be reduced (i.e., reducing τ) so that the bit rate overhead and distortion can be kept to the minimum. On the other hand, when there is a need to accommodate more data, the video encoding process is made to code more coding blocks by using the smallest block size (i.e., increasing τ). Both intra prediction mode and MVP of SHVC video are jointly utilized to embed data. The data embedding opportunity is hence increased by jointly utilizing both intra prediction mode and MVP. The encoder calculates the prediction errors based on the original and the predicted values after the predictive syntax elements are modified to match the payload bits. Therefore, the reconstructed video is a close approximation of the original video and the perceptual quality of the video is maintained without error-drift. Experimental results suggest that the achieved embedding capacity is encouraging while the video quality and bit rate are maintained. A trade-off between the increase in embedding capacity and bit rate overhead can be achieved by tuning the threshold τ depending on the application of interest.

As future work, we aim to explore different venues for data embedding in SHVC and investigate into their impact on the video quality as well as the bit rate overhead. We will also explore potential joint utilization of the proposed and the conventional data embedding methods.

Author Contributions: L.P. contributed to formulating the ideas and concepts data embedding in SHVC. She also implemented the ideas, conducted experiments, and compiled this article. K.W. contributed to the formulation of idea and experiment setup. He also supervised the overall compilation of this article and contributed to reviewing and editing, in particular, the structure of this article. Y.T. contributed to idea formulation, technical input to the coding standard of SHVC, the setup of the experiment, as well as validation of the experimental results. He also contributed to writing. S.R. contributed to the supervision of this project, fund acquisition, and the technical input to the coding standard of HEVC/SHVC. He also contributed to the writing as well as comparative analysis. All authors have read and agreed to the published version of the manuscript.

Funding: The work of S. Rahardja was supported in part by the Overseas Expertise Introduction Project for Discipline Innovation (111 project: B18041).

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Datasets analyzed in this study were retrieved from [35,36]. However, the ftp link to [35] is longer accessed by public. Ref. [35] Universität-Hannover. Test Sequence. ftp://ftp.tnt.uni-hannover.de/testsequences/. Ref. [36] XIPH.ORG. Derf's collection. https://media.xiph.org/video/derf.

Conflicts of Interest: The authors declare that they have no known competing financial interest or personal relationship that could have appeared to influence the work reported in this paper.

References

1. Tew, Y.; Wong, K. An Overview of Information Hiding in H.264/AVC Compressed Video. *IEEE Trans. Circuits Syst. Video Technol.* **2014**, *24*, 305–319. [CrossRef]
2. Hao, Y.; Ngo, C.; Huet, B. Neighbourhood Structure Preserving Cross-Modal Embedding for Video Hyperlinking. *IEEE Trans. Multimed.* **2020**, *22*, 188–200. [CrossRef]
3. Chung, B.; Yim, C. Bi-Sequential Video Error Concealment Method Using Adaptive Homography-Based Registration. *IEEE Trans. Circuits Syst. Video Technol.* **2020**, *30*, 1535–1549. [CrossRef]
4. Jiang, L.; Ma, X.; Chen, S.; Bailey, J.; Jiang, Y.G. Black-Box Adversarial Attacks on Video Recognition Models. In *Proceedings of the 27th ACM International Conference on Multimedia MM '19*; Association for Computing Machinery: New York, NY, USA, 2019; pp. 864–872.
5. Corcoran, P.; Cucos, A. Techniques for securing multimedia content in consumer electronic appliances using biometric signatures. *IEEE Trans. Consum. Electron.* **2005**, *51*, 545–551. [CrossRef]
6. Tew, Y.; Wong, K.; Phan, R.C.W.; Ngan, K.N. Separable authentication in encrypted hevc video. *Multimed. Tools Appl.* **2018**, *77*, 24165–24184. [CrossRef]
7. Perazzone, J.B.; Yu, P.L.; Sadler, B.M.; Blum, R.S. Cryptographic Side-Channel Signaling and Authentication via Fingerprint Embedding. *IEEE Trans. Inf. Forensics Secur.* **2018**, *13*, 2216–2225. [CrossRef]
8. Rao, K.; Kim, D.N.; Hwang, J.J. *Video Coding Standards*; Springer: Dordrecht, The Netherlands, 2014; pp. 51–97.
9. 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]
10. Boyce, J.M.; Ye, Y.; Chen, J.; Ramasubramonian, A.K. Overview of SHVC: Scalable Extensions of the High Efficiency Video Coding Standard. *IEEE Trans. Circuits Syst. Video Technol.* **2016**, *26*, 20–34. [CrossRef]
11. Schwarz, H.; Marpe, D.; Wiegand, T. Overview of the Scalable Video Coding Extension of the H.264/AVC Standard. *IEEE Trans. Circuits Syst. Video Technol.* **2007**, *17*, 1103–1120. [CrossRef]
12. Sikora, T. MPEG digital video-coding standards. *IEEE Signal Process. Mag.* **1997**, *14*, 82–100. [CrossRef]
13. 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*, 331–337. [CrossRef]
14. Sheng, Q.; Wang, R.; Pei, A.; Wang, B. An Information Hiding Algorithm for HEVC Based on Differences of Intra Prediction Modes. In *Cloud Computing and Security*; Sun, X., Liu, A., Chao, H.C., Bertino, E., Eds.; Springer: Cham, Switzerland, 2016; pp. 63–74.
15. Mareen, H.; De Praeter, J.; Van Wallendael, G.; Lambert, P. A Novel Video Watermarking Approach Based on Implicit Distortions. *IEEE Trans. Consum. Electron.* **2018**, *64*, 250–258. [CrossRef]
16. Saberi, Y.; Ramezanpour, M.; Khorsand, R. An efficient data hiding method using the intra prediction modes in HEVC. *Multimed. Tools Appl.* **2020**, *79*, 33279–33302. [CrossRef]
17. Nguyen, C.; Tay, D.B.H.; Deng, G. A Fast Watermarking System for H.264/AVC Video. In Proceedings of the APCCAS 2006–2006 IEEE Asia Pacific Conference on Circuits and Systems, Singapore, 4–7 December 2006; pp. 81–84.
18. Fang, D.Y.; Chang, L.W. Data hiding for digital video with phase of motion vector. In Proceedings of the IEEE International Symposium on Circuits and Systems, Kos, Greece, 21–24 May 2006; pp. 1422–1425.
19. Aly, H.A. Data Hiding in Motion Vectors of Compressed Video Based on Their Associated Prediction Error. *IEEE Trans. Inf. Forensics Secur.* **2011**, *6*, 14–18. [CrossRef]
20. Van, L.P.; Praeter, J.D.; Wallendael, G.V.; Cock, J.D.; de Walle, R.V. Out-of-the-loop information hiding for HEVC video. In Proceedings of the IEEE International Conference on Image Processing (ICIP), Quebec City, QC, Canada, 27–30 September 2015; pp. 3610–3614.
21. Pang, L.; Wong, K. A Data Embedding Technique for Spatial Scalable Coded Video Using Motion Vector Predictor. In Proceedings of the IEEE International Conference on Image Processing (ICIP), Taipei, Taiwan, 22–25 September 2019; pp. 4050–4054.
22. Tew, Y.; Wong, K. Information hiding in HEVC standard using adaptive coding block size decision. In Proceedings of the IEEE International Conference on Image Processing (ICIP), Paris, France, 27–30 October 2014; pp. 5502–5506.
23. Shanableh, T. Altering split decisions of coding units for message embedding in HEVC. *Multimed. Tools Appl.* **2018**, *77*, 8939–8953. [CrossRef]
24. 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]
25. Buhari, A.M.; Ling, H.C.; Baskaran, V.M.; Wong, K. Fast watermarking scheme for real-time spatial scalable video coding. *Signal Process. Image Commun.* **2016**, *47*, 86–95. [CrossRef]
26. Singh, R.; Nigam, S.; Singh, A.K.; Elhoseny, M. On Wavelet Domain Video Watermarking Techniques. In *Intelligent Wavelet Based Techniques for Advanced Multimedia Applications*; Springer: Cham, Switzerland, 2020; pp. 65–76.
27. Vybornova, Y. A New Watermarking Method for Video Authentication with Tamper Localization. In *Computer Vision and Graphics*; Chmielewski, L.J., Kozera, R., Orłowski, A., Eds.; Springer: Cham, Switzerland, 2020; pp. 201–213.
28. Shanableh, T. Matrix encoding for data hiding using multilayer video coding and transcoding solutions. *Signal Process. Image Commun.* **2012**, *27*, 1025–1034. [CrossRef]
29. Amiri, M.; Amiri, A.; Meghdadi, M. HVS-based scalable video watermarking. *Multimed. Syst.* **2019**, *25*, 1–19.

30. Pang, L.; Wong, K.; Liong, S.T. Data embedding in scalable coded video. In Proceedings of the Asia-Pacific Signal and Information Processing Association Annual Summit and Conference (APSIPA), Kuala Lumpur, Malaysia, 12–15 December 2017; pp. 1190–1194.
31. Sun, Y.; Wang, J.; Huang, H.; Chen, Q. Research on scalable video watermarking algorithm based on H.264 compressed domain. *Optik* **2020**, *227*, 165911. [CrossRef]
32. Huo, W.; Zhu, Y.; Chen, H. A Controllable Error-Drift Elimination Scheme for Watermarking Algorithm in H.264/AVC Stream. *IEEE Signal Process. Lett.* **2011**, *18*, 535–538. [CrossRef]
33. SHM-12.0. Available online: https://hevc.hhi.fraunhofer.de/svn/svn_SHVCSoftware/ (accessed on 1 February 2021).
34. Seregin, V.; He, Y. Common SHM Test Conditions and Software Reference Configurations, Document: JCTVC-P1009. In Proceedings of the JCT-VC 16th Meeting, San Jose, CA, USA, 9–17 January 2014.
35. Universität-Hannover. Test Sequence. Available online: ftp://ftp.tnt.uni-hannover.de/testsequences/ (accessed on 15 November 2014).
36. XIPH.ORG. Derf's Collection. Available online: https://media.xiph.org/video/derf (accessed on 15 November 2014).
37. Wang, Z.; Bovik, A.C.; Sheikh, H.R.; Simoncelli, E.P. Image quality assessment: From error visibility to structural similarity. *IEEE Trans. Image Process.* **2004**, *13*, 600–612. [CrossRef]
38. Bjøntegaard, G. Calculation of average PSNR differences between RD-Curves. In Proceedings of the ITU-T Video Coding Experts Group (VCEG) Thirteenth Meeting, Austin, TX, USA, 2–4 April 2001.
39. Noorkami, M.; Mersereau, R.M. Digital Video Watermarking in P-Frames With Controlled Video Bit-Rate Increase. *IEEE Trans. Inf. Forensics Secur.* **2008**, *3*, 441–455. [CrossRef]
40. Hu, Y.; Zhang, C.; Su, Y. Information Hiding Based on Intra Prediction Modes for H.264/AVC. In Proceedings of the IEEE International Conference on Multimedia and Expo, Beijing, China, 2–5 July 2007; pp. 1231–1234.
41. Konyar, M.Z.; Akbulut, O.; Öztürk, S. Matrix encoding-based high-capacity and high-fidelity reversible data hiding in HEVC. *Signal Image Video Process.* **2020**, *14*, 897–905. [CrossRef]
42. Shen, L.; Zhang, Z.; Liu, Z. Adaptive inter-mode decision for HEVC jointly utilizing inter-level and spatiotemporal correlations. *IEEE Trans. Circuits Syst. Video Technol.* **2014**, *24*, 1709–1722. [CrossRef]
43. Pang, L.; Wong, K.; Ito, R. Merge Mode-based Data Embedding in SHVC Compressed Video. In Proceedings of the International Symposium on Intelligent Signal Processing and Communication Systems (ISPACS), Taipei, Taiwan, 3–6 December 2019; pp. 1–2.
44. Chen, J.; Boyce, J.; Ye, Y.; Hannuksela, M. SHVC Test Model 10 (SHM 10) Introduction and Encoder Description, Document: JCTVC-U1007. In Proceedings of the JCT-VC 21st Meeting, Warsaw, Poland, 19–26 June 2015.