Multimedia Tools and Applications (2021) 80:13121–13142
https://doi.org/10.1007/s11042-020-10359-w

# CUPSEED - A combined use of prediction syntax elements to embed data in SHVC video

**LieLin Pang[1] · Yiqi Tew[2] · KokSheik Wong[3,4] · Mohamad Nizam Bin Ayub[1]**

Received: 14 June 2020 / Revised: 5 October 2020 / Accepted: 22 December 2020 /
Published online: 13 January 2021


**Abstract**
With the rapid advancement in digital technologies, video rises to become one of the most effective communication tools that continues to gain popularity and importance. As a result, various proposals are put forward to manage videos, and one of them is data embedding. Essentially, data embedding inserts data into the video to serve a specific purpose, including proof of ownership via watermark, covert communication in steganography, and authentication via fragile watermark. However, most conventional methods embed data by using only one type of syntax element defined in the video coding standard, which may suffer from large bit rate overhead, quality degradation, or low payload. Therefore, this work aims to explore the combined use of multiple prediction syntax elements in SHVC for the purpose of data embedding. Specifically, the intra prediction mode, motion vector predictor, motion vector difference, merge mode and coding block structure are collectively manipulated to embed data. The experimental results demonstrate that, in comparison to the conventional single-venue data embedding methods, the combined use of prediction syntax elements can achieve higher payload while preserving the perceptual quality with minimal bit rate variation. In the best case scenario, a total of 556.1 kbps is embedded into the video sequence *PartyScene* with a drop of 0.15 dB in PSNR while experiencing a bit rate overhead of 7.4% when all prediction syntax elements are utilized altogether. A recommendation is then put forward to choose specific types of syntax element for data embedding based on the characteristics of the video.

**Keywords** CUPSEED · Data embedding · Prediction syntax element · SHVC

## 1 Introduction

With the advancement in digital and communication technologies, the ability to record, edit and share high quality video through the internet have grown tremendously over the years. Enjoying video on demand (VOD), participating in online meeting, attending webinar, learning via lecture recording have become common sights around the world. Recent

✉ KokSheik Wong
wong.koksheik@monash.edu

Extended author information available on the last page of the article.

pandemic due to the situation of Covid-19 further accelerated the adoption of online classroom for teaching and learning purposes, and more lecture / tutorial videos are recorded and shared around the world.

As video becomes an increasingly important tool for communication, productivity and now education, there is an urgent need for new approaches to effectively manage huge volume of video files. Some of the common problems include tampered video content, unauthorized / illegal use of video content, and various threats to businesses for storing sensitive video contents using cloud storage. Tew et al. embeds authentication code in both encrypted and decrypted (plainsight) videos to verify the integrity of a video [21]. On the other hand, the computer / cloud administrator may need additional information to handle encrypted video (e.g., how to transcode or compress the encrypted video), linking related videos, managing ownership as well as other legal aspects of the video, to name a few of the tasks. For the aforementioned purposes, data embedding, which inserts some data into the video, can serve as one of the solutions to manage videos.

Regardless of the application of interest, it is crucial to minimize the degradation in perceptual quality caused by embedding data into the video. The requirement on quality is particularly important in steganography, which aims to secretly embed a large payload [2]. Other applications have different requirements, for example, the survival of the embedded data as in the application of watermarking [20], quick data extraction for continuous authentication [25], and reversibility for handling sensitive content or rare artwork [9]. However, increased payload usually leads to image distortion or quality degradation. Abdulla et al. propose to decompose pixel intensity values into 16 bit-planes to increase the payload. The proposed scheme achieves a good trade-off between payload and image quality [1]. In addition, bit-plane(s) mapping technique is invented to increase similarity between the binary secret image and the LSB plane of the cover image for reducing changes due to data embedding while maintaining payload [3]. On the other hand, Konyar et al. utilize a matrix encoding-based approach to achieve the same goal [10]. The strengths and weaknesses of the matrix encoding are identified to achieve a good trade-off among payload, increase in bit rate, and video quality.

In general, data embedding in video takes place either in the spatial, temporal or transformed domain. Data can be embedded into difference venues of a video sequence during video encoding. For this work, we focus on Scalable High Efficiency Video Coding (SHVC) [5], which is a recent video coding standard that enables a video to be encoded in multiple layers within a single bit stream. It supports video communication over varying network and bandwidth conditions, video of different quality, as well as devices with different processing capabilities. It also allows a decoder to decode different number of layers (i.e., sub streams), depending on the aforementioned conditions, hence gracefully degrading the video quality.

In this work, we propose a data embedding technique that utilizes multiple prediction syntax elements (PSEs). Arguably, prediction is one of the most important ingredients in video coding, because this technique is able to effectively remove spatial and temporal redundancies. Specifically, prediction syntax elements, which include intra prediction mode (IPM), motion vector predictor (MVP), motion vector difference (MVD), merge mode (MRG) and block structure (BSZ), are manipulated in a combined manner to embed data. The main contributions of this work are:

1. comprehensive analysis on the impact of using different SHVC prediction elements to embed data;

2. improve payload (i.e., number of bits that can be embedded) by utilizing multiple PSEs without compromising video quality while minimizing bit rate variation, and;
3. propose a data embedding technique - CUPSEED, which automatically selects different prediction elements for data embedding.

This paper is organized as follows: Section 2 briefly reviews the existing data embedding methods. Section 3 provides an overview of the coding block structure and prediction modes in SHVC. The proposed combined use of PSEs for data embedding is then detailed in Section 4. The experimental results and discussion are presented in Section 5. Finally, Section 6 concludes this paper.

## 2 Related works

Many methods are proposed to embed data into the existing video coding standards, including MPEG-2, MPEG-4, H.264/AVC, as well as HEVC, for the goal of providing additional managerial features [20]. The embedding process can be executed at various stages, including during prediction, motion estimation, transformation, quantization, as well as entropy encoding. For the purpose of this work, our review focuses on the conventional methods that are based on the prediction syntax elements.

Specifically, the number of intra prediction modes has increased from 9 in H.264/AVC to 35 in HEVC and SHVC, leading to more opportunities for data embedding. For example, Wang et al. [24] and Xu et al. [27] map payload data to IPMs of the smallest prediction blocks (PBs) in a HEVC coded video. Sheng et al. then utilize the difference of two consecutive intra prediction modes (i.e., directions) or a pair of continuous planar or DC mode of the smallest PBs to embed data [17]. However, for these methods, the number of PBs that can be utilized for embedding is reduced when other prediction mode or block size has better rate distortion cost ($RDC$).

For motion vectors, Nguyen et al. [11] and Xu et al. [28] propose to embed data into a H.264/AVC video during inter prediction by exploiting the magnitude of both the horizontal and vertical components of MV. On the other hand, the difference between the phase angles for MV pairs are manipulated by Fang et al. [8] to embed data. When the phase angle difference does not satisfy the embedding condition, one of the MVs is replaced by a qualified MV. Aly then associates data to MVs with high prediction error in MPEG-2 compressed video [4]. It is designed with the argument that distortion caused by data embedding will be less obvious when the prediction error is large. However, selecting MVs with high prediction error does not always lead to minimum quality degradation.

In addition to the aforementioned methods, there are methods based on other prediction syntax elements. Van et al. [23] manipulate MVD and transformed coefficients to embed data into a HEVC video. Similarly, Yang et al. proposed to embed data by using selected MV components in the smallest PUs for HEVC [29]. Instead of dealing with MV/MVD, Tew et al. manipulate PB size based on some predefined mapping rules and nonzero AC coefficients in HEVC coded video to embed data [19]. Moreover, Shanableh associates the data to the split flag, which is utilized in HEVC to decide whether a block is split into smaller blocks. Specifically, a weighting model is introduced to predict split decision of a block. If the data bit to be embedded is '1', both flags must be identical, otherwise the coded split flag must be different from the predicted flag [16]. The comparison of embedding capacity for different partitioning levels can be found in [30]. It is noteworthy that the embedding

capacity for block partitioning-based approach depends on the texture of the video. Specifically, a video frame with more smooth regions tends to be coded by using larger blocks hence such frame usually carries lower payload, and vice versa.

Undoubtedly, the conventional methods proposed for HEVC can be adopted to SHVC. However, there are additional features in SHVC that can exploited to improve data embedding approach. For instance, the payload can be increased by using multi-coded layers. In addition, video quality degradation introduced by data embedding can be minimized by using inter layer prediction. Buhari et al. embedded data into the DCT coefficients of high textured blocks of H.264/SVC [7]. Pang et al. put forward an error compensation embedding technique during intra prediction [13]. In [14], a threshold is introduced to split block to improve the payload when associating parity bits of the MVP indices with data bits. Recently, Pang et al. proposed to embed data into SHVC compressed video by manipulating the merge candidate blocks [15], but only one PSE (venue) is considered. To the best of our knowledge, there is no prior work that manages data embedding by using multiple venues in SHVC compressed video. Considering the advantages and potentials of embedding data into scalable coded video, we put forward some recommendations for managing the process of embedding data into SHVC by using multiple PSEs.

## 3 Overview of coding block structure and selection of prediction mode

In order to keep the paper self-contained, we review the coding block structure and selection of prediction mode in SHVC.

### 3.1 Coding block structure

Quad-tree based block partitioning has significantly improved the coding efficiency for visual content. In SHVC, a video frame is first partitioned into blocks each of size $64 \times 64$ pixels. Subsequently, each block can potentially be split into four smaller blocks each of size $32 \times 32$. The partitioning process continues until the smallest block size (i.e., $4 \times 8$ and $8 \times 4$ for inter-picture prediction, or $4 \times 4$ for intra-picture prediction) is reached. Altogether, there

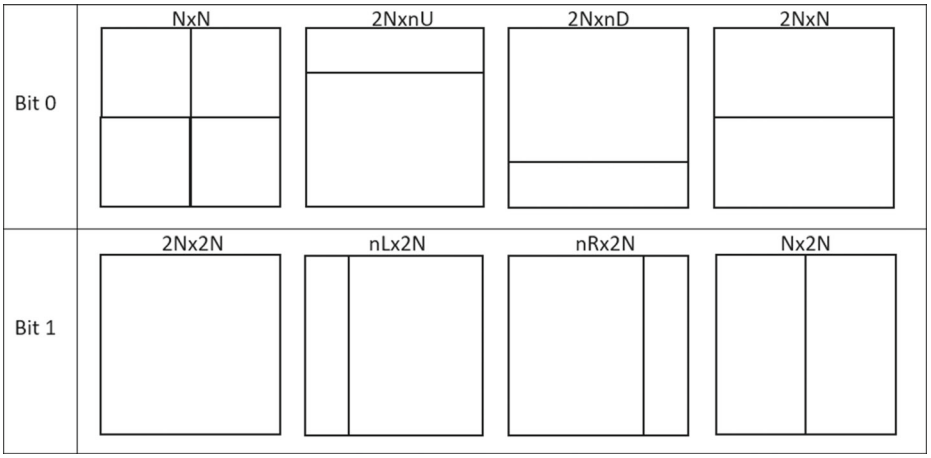


**Fig. 1** Coding block structure and payload bits representation

are eight different coding block structures as illustrated in Fig. 1. In essence, the coding block structure determines the internal partitioning of a 64 × 64 block. In general, a homogeneous or smooth region is coded by using a larger block size while a complex/textured region is coded by using a combination of smaller blocks.

Here, the rate distortion optimizer (RDO) determines which particular coding structure and block size to use. The goal is to achieve a good trade-off between the visual quality and the number of bits spent on coding the block. Specifically, the RDO computes the function $RDC = D + \lambda R$, where $D$ is the distortion between the original input and reconstructed signals, while $R$ is the compression rate which represents the number of bits spent on coding. The parameter $\lambda$ is the Lagrange multiplier used for Lagrangian optimization. Let $d$ denote the partitioning depth level, where $d = 0$, 1, 2 and 3 correspond to block size of $N = 64$, 32, 16 and 8, respectively. The $RDC$ calculated for each depth level $d$ is compared with the accumulated $RDC$ for the split blocks at the next depth level, i.e., $d + 1$. For example, for maximum depth $d_{\max} = 3$, the accumulated cost at depth level $d = 3$ ($D_3$) is compared with the cost at depth level $d = 2$ ($D_2$). Then the cost at $D_2$ is compared to the cost at depth level $d = 1$ ($D_1$). The comparison process continues until all block sizes are compared and coded, which ends at the root of the quad-tree, i.e., $d = 0$ ($D_0$). Let $RDC_{best}$ be the $RDC$ with the optimal cost. When the $RDC$ for a larger block (i.e., denoted by $p_d$ for level $d$) is more than the accumulated $RDC$ of split blocks (denoted by $p_{d+1}$), the split blocks are coded instead of the larger block. Here,

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

where $n_d$ is the total number of split blocks at level $d$.

### 3.2 Prediction mode

This section briefly reviews the prediction modes which are utilized in our work to embed data.

#### 3.2.1 Intra prediction mode

In spatially scalable video coding, video is coded into multiple layers, including one base layer (BL) at the lowest resolution, and one or more enhancement layers (ELs) with higher resolutions. Recall that the spatial collocated blocks in different layers correspond to the same area of a frame. Hence, the BL (or a lower layer) of a scalable coded video can be used as a reference layer for predicting the video frame in EL. In addition, within a video frame, neighboring blocks usually have similar textures and they are highly correlated. Consequently, during intra-picture prediction, the current block is predicted by using the decoded spatial-predicted information of its neighboring blocks and/or collocated block in the reference layer. Specifically, there are 35 IPMs for predicting the current block, which includes planar (mode 0), DC (mode 1), and 33 directional modes (mode 2 $\sim$ 34) [6].

#### 3.2.2 MVP and MVD

During inter-picture (a.k.a motion compensation) prediction, temporal redundancy between adjacent video frames is removed by motion prediction. Within a video frame, the neighboring blocks and collocated blocks in adjacent frames (i.e., used as reference picture) are likely

corresponding to the same moving object with similar motion. Hence, a PB may use the motion information of its neighboring blocks and/or collocated blocks in adjacent frames to remove temporal redundancy. Several MVP candidates from spatial and temporal neighbors blocks are determined during the advanced motion vector prediction (AMVP) process [6]. For the example shown in Fig. 2, the potential MVP candidates for block $X$ include $A_0$, $A_1$, $B_0$, $B_1$ and $B_2$ from the neighboring blocks, and $C_0$ and $C_1$ from the collocated blocks in the reference picture. In order to reduce bit rate required to code the predicted MV, the index of the best MVP candidate, i.e., the one having the optimal cost, will be selected and encoded. Next, MVD is calculated as $MVD_i = MV_i - MVP_i$, where $i \in \{x, y\}$ for the horizontal ($x$) and vertical ($y$) components of MV, respectively. Essentially, MVD captures the difference between the MV of the best MVP candidate and the predicted MV of the current block. Both MVP index of the best candidate and MVD are coded and transmitted to the decoder.

### 3.2.3 Merge mode candidate

In video coding, adjacent blocks usually contain objects with similar motion, which can be predicted by using the same MV. Hence, in addition to the inter prediction mode using AMVP, merge mode is also utilized for deriving motion information from spatially or temporally neighboring blocks and/or reference frame. The list of MV candidates in merge mode is similar to the MVP candidates (which comprises of five spatial candidates and two temporal candidates) as depicted in Fig. 2.

Each PU can be predicted by using an intra- or inter-picture prediction mode. For PU coded in inter-picture prediction mode, either the predicted motion vector (MV) or MV of neighboring block/frame (i.e., merge mode) is used for deriving the motion data. The selection of prediction mode is based on the best RD cost. Let $\Psi$ be the set of all possible intra and inter prediction modes, and let $\psi \in \Psi$ be the coding mode applied on block $b_i$. The coding mode $\psi_i$ is selected according to

$$\psi_i = \underset{\psi \in \Psi}{\operatorname{argmin}}\, D_i(\psi) + \lambda R_i(\psi), \tag{2}$$

where the distortion $D_i(\psi)$ represents the sum of squared differences between the original block $b_i$ and the reconstructed block $b_i'$ (i.e., the result of coding $b_i$ by using mode $\psi$). The term $R_i(\psi)$ represents the number of bits spent on coding the blocks $b_i$ by using the coding mode $\psi$. It includes the number of bits required for signaling the coding mode and the associated side information (e.g., MVs, reference indices, IPM and coding modes for all

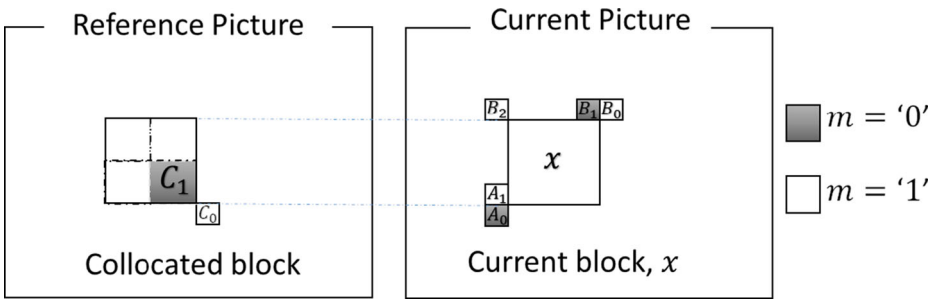


**Fig. 2** Merge candidates mapping

partitioned blocks of $b_i$), as well as the number of bits spent on transmitting the transformed coefficient levels to store the residual signal [12].

## 4 Proposed data embedding method

The advantage of the combined use of all prediction syntax elements for data embedding is that it allows some form of optimization, where the best mode can be selected for data embedding purposes. In contrast, when restricting data embedding to a single prediction mode (venue), there is no guarantee that a payload bit can be embedded successfully into a particular block. For example, suppose only IPM is manipulated for data embedding. After embedding data into a particular block, a potential outcome is that IPM costs more than MV for coding the block in question. In such a situation, IPM will not be employed and the payload bit fails to be embedded into the block. From another perspective, the combined use of MVP, MVD, IPM and MRG also increases the payload. Specifically, each block holds one bit, unless it is identified to be a skipped block.

### 4.1 Combined use of PSEs for data embedding

In this work, the intra and inter prediction PSEs in all scalable coded layers are utilized in a combined manner for data embedding purposes. Due to the complexity of the SHVC codec,

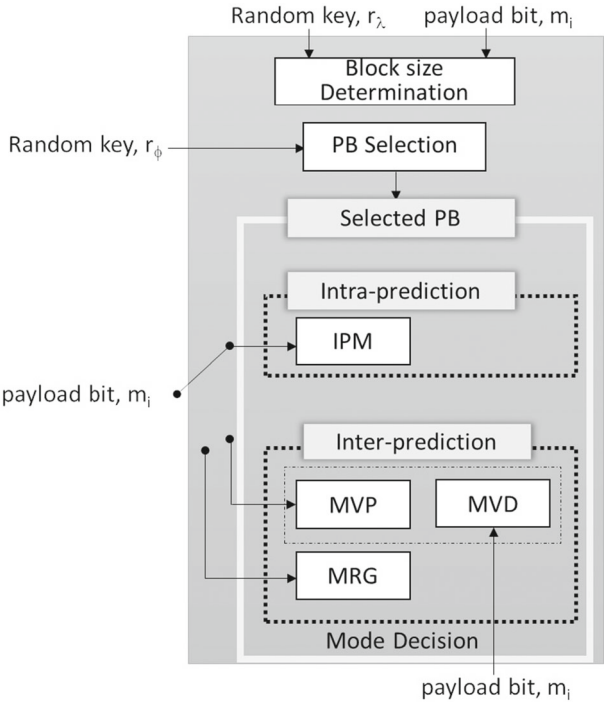


**Fig. 3** Flow of embedding process

the quad-tree splitting and pruning process in CTUs has to be managed well to ensure that each data bit is embedded accordingly. The flow of processes is illustrated in Fig. 3. Let $M = \{m_i\}_1^N$ be the payload data, where $m_i \in \{0, 1\}$ and $N$ is the length of $M$. Then, data embedding takes place at five venues, namely, BSZ, IPM, MVP, MVD, and MRG.

First, the data bit $m_i$ is mapped to the PB by selecting the best block structure which represents $m_i$. Specifically, all coding block structures mapped to $m_i$ are evaluated by using all applicable prediction modes, and the one achieving $RDC_{best}$ is coded. One of the possible mapping rules is shown in Fig. 1. Data embedding using BSZ is applicable to the I, P and B-slices. For the case of I-slice, all PBs are encoded by using $N \times N$ block for $m_i = 0$, or $2N \times 2N$ block for $m_i = 1$, respectively. On the other hand, for the case of P- and B-slices, there are more coding block structure for consideration. In both cases, the coding block structure are divided into two groups, where one group represents '0' and the other represents '1'. In particular, the first group includes blocks of type $N \times N$, $2N \times nU$, $2N \times nD$ and $2N \times N$, which is utilized to represent payload bit '0'. On the other hand, the other group includes blocks of type $2N \times 2N$, $nL \times 2N$, $nR \times 2N$ and $N \times 2N$, which is utilized to represent payload bit '1'. The block associated with the data (i.e., $m_i$) having the best $RDC$ will be selected for coding. Here, random keys $\kappa_\lambda$ can be utilized for selecting/skipping candidate blocks for embedding.

The prediction modes of IPM, MVP and MRG are manipulated to embed payload bit $m_i$ during the encoding process. Here, another key, $\kappa_\phi$ can be utilized to select/skip blocks for data embedding. Basically, a video frame is partitioned into a number of coding tree units (i.e., $64 \times 64$). For each CTU (i.e., at $D_0$), the coding block is further split into smaller blocks. To ease the presentation, let the term *parent block* refer to the block before splitting as illustrated in Fig. 4. For each block, the encoder predicts the block by using all possible prediction modes, and for each prediction mode, all possible coding block structures (i.e., partitioning) are considered. Specifically, RDO determines the best coding structure for each prediction mode. Then, comparison of $RDC$ for different prediction modes is performed, and the one achieving $RDC_{best}$ is retained for subsequent comparison (i.e., line 3 to 6, and 8 to 11 in Algorithm 3). Next, each of these blocks (i.e., at $D_1$) is further split into smaller blocks (i.e., $D_2$). The same prediction and splitting processes (i.e., line 13 to 23 in Algorithm 3) are performed for each block, and the processes is repeated until the smallest block size is reached. When the best prediction mode in the smallest blocks (e.g., those

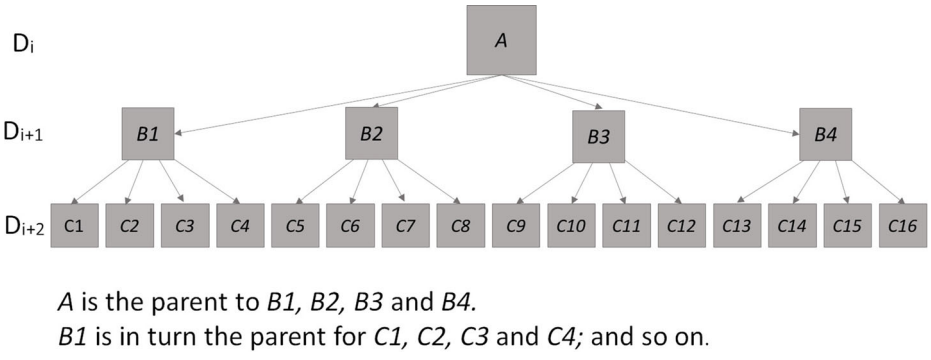


**Fig. 4** Illustration of parent block and its children block

labelled with $C$ at the bottom of Fig. 4) is decided, the $RDC$ for the sibling split blocks are accumulated and compared with the $RDC$ for the parent block. As an example, the sum of $RDC$ of $C1$, $C2$, $C3$ and $C4$ is compared with $RDC$ of $B1$. The prediction-splitting combination that yields $RDC_{best}$ is selected to code the 64 × 64 block. The comparison process starts from the bottom (i.e., smallest block size), then moves up one level at a time, and eventually to the root until the entire block-partitioning process in the quad-tree is completed (i.e., at $D_0$). At each partitioning depth level $d$, the prediction modes compete among each other, and the one with $RDC_{best}$ is identified. Hence, the combined used of prediction modes of IPM, MVP and MRG can increase the opportunities for data embedding.

**Algorithm 1** Embed data bit into MRG.

**input** : $RDC_{best}$, payload data $M$, $BestMode$
**output**: $RDC_{best}$, $BestMode$
get payload bit $m_i$ from $M$
Construct a list of merge candidate, $C$ based on payload bit $m_i$
$RDC_{MRG} \leftarrow MAX_{DOUBLE}$
**for** $c_i$ *in* $C$ **do**
    Perform motion compensation and residual prediction
    Calculate $RDC_i$ for $c_i$
    **if** $RDC_i < RDC_{MRG}$ **then**
        $RDC_{MRG} \leftarrow RDC_i$
        $mergeCand \leftarrow c_i$
    **end**
**end**
**if** $RDC_{MRG} < RDC_{best}$ **then**
    $RDC_{best} \leftarrow RDC_{MRG}$
    $BestMode \leftarrow MergeMode$
**end**

**Algorithm 2** Embed data bits into MVP and MVD.

**input** : $RDC_{best}$, payload data $M$, $BestMode$
**output**: $RDC_{best}$, $BestMode$
get payload bit $m_i$ from $M$
Derive MVP candidates from spatial and temporal candidate list
**if** $MVP\ exist$ **then**
    select MVP with MVP_index that matches $m_i$
    get payload bit $m_i$ from $M$
**end**
$MVD \leftarrow MV - MVP$
Check and get the longest magnitude of $MVD_i$, $i$ is horizontal or vertical component of MVD
**if** $m_i \neq parity(MVD_i)$ **then**
    **if** $MVD_i \geq 0$ **then**
        $MVD_i \leftarrow MVD_i - 1$
    **else**
        $MVD_i \leftarrow MVD_i + 1$
    **end**
**end**
Perform motion compensation and residual prediction
Check $RDC_{MV}$ for inter prediction mode
**if** $RDC_{MV} < RDC_{best}$ **then**
    $RDC_{best} \leftarrow RDC_{MV}$
    $BestMode \leftarrow InterMode$
**end**
Algorithm 1

**Algorithm 3** Combined used of PSEs for data embedding.

```
CUPSEED (CU, d_i, M)
get m_i from M
if m_i = 1 then
    for block structure ∈ 2N × 2N , N × 2N, nL × 2N, nR × 2N do
        Algorithm 2 /* Embed data bits into inter mode */
    end
    Embed data bit into IPM using 2N × 2N
else
    for block structure ∈ N × N , 2N × N, 2N × nU, 2N × nD do
        Algorithm 2
    end
    Embed data bit into IPM using N × N
end
if d_i < d_max then
    split CU into cu_j, j ∈ 1..4
    for j ←1 to 4 do
        CUPSEED (cu_j, d_{i+1}, M)
    end
    calculate accumulated RDC_{d_{i+1}}
    if RDC_{d_{i+1}} < RDC_best then
        RDC_best ← RDC_{d_{i+1}}
        CU ← cu_j, j ∈ 1..4
    end
end
return;
```

During motion compensation prediction, we consider both MVP and MVD to increase payload. In particular, attempts are made to map $m_i$ to indices of MVP for a particular PB. The MVP candidates are depicted in Fig. 2. For each PB, $m_i$ is mapped to either the best or first runner-up MVP index. Then, MVD is calculated, and the magnitude for the horizontal and vertical components are compared. Note that smaller MVD requires less bits for coding, and vice versa. Hence, if the parity bit of the MVD component with the larger magnitude differs from $m_i$, the magnitude of the MVD is reduced by one as depicted in Fig. 5. The

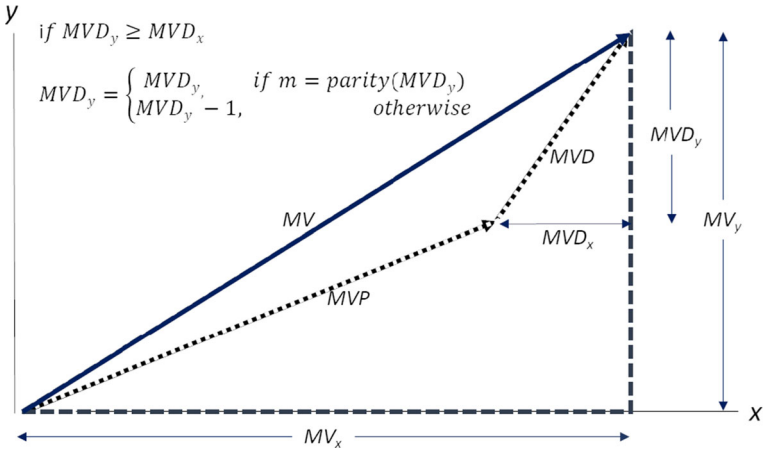


**Fig. 5** Example of payload bit representation using MVD

prediction information for the best PB (denoted by $RDC_{MV}$) is then selected for subsequent comparison.

The block merging candidate selection decision is manipulated to embed data based on some pre-defined mapping rules. Again, merge candidates are divided into two groups, where one is associated with bit '0' and the other associated with bit '1'. An example of mapping rules applied to merge candidates is shown in Fig. 2. To determine the merge candidate for the current block (CurrBlk), $m_i$ is mapped to the merge candidates associated to the payload bit as illustrated in Algorithm 1. The prediction information of merge candidate block with the best cost (denoted by $RDC_{MRG}$) is selected. The $RDC_{MRG}$ is then compared with $RDC_{best}$. The one with the lower cost is selected for subsequent comparison.

For IPM, only the prediction directional modes are considered for data embedding. This is to ensure that the bit rate overhead is kept to the minimum. During intra picture prediction, the parity bit of the prediction mode is manipulated for data embedding. When the best predicted direction in BL has an index that differs from the data bit $m_i$, it is modified by increasing the prediction direction mode by unity. In contrast, when the parity bit differs from $m_i$ in EL, the prediction direction mode is reduced by unity. For the boundary cases, the opposite prediction direction is applied. In such way, any changes or noise introduced to the prediction mode in BL can be compensated in EL when changes are made towards the opposite prediction direction. Specifically, the angular difference between the best and selected modes is small, which is $\sim 5.6°$ [24]. When a video is coded in more than two layers, the error introduced in the reference layer (i.e., BL) can be compensated in the EL to reduce the bit rate variation [13]. The prediction mode that yields the best cost (denoted by $RDC_{IPM}$) is then compared with $RDC_{best}$. The one with the lower cost is selected for coding. Note that only the smallest coding blocks are utilized for data embedding in IPM, MVP and MVD in order to minimize bit rate variation and to preserve video quality. In addition, $RDO$ plays an important role in determining the prediction mode for coding. For example, suppose the original best prediction mode for a PB is IPM. When embedding $m_i$ into that PB, $RDC$ for all prediction modes associated with $m_i$ are evaluated. Suppose the merge mode has $RDC_{best}$. In such a situation, merge mode will be coded instead of IPM.

### 4.2 Data extraction

During decoding, the embedded data can be extracted from each PSE in the PBs identified by using $\kappa_\lambda$ and $\kappa_\phi$ (i.e., to determine which one is skipped / selected), which are only known by the authorized parties. First, the embedded data is retrieved by checking the *parity* of the intra prediction directional mode in the identified embedding block. Recall that when the parity bit of the MVD component with the larger magnitude differs from the payload bit, the magnitude of the MVD is reduced by unity. Hence, for extracting the embedded data from MVP and MVD, the *parity* of the MVP (i.e., indices) and MVD components with the same or larger MV magnitude for the identified PBs are extracted. As an illustration, Fig. 5 shows the MV with MVP and MVD components. The magnitude of $MVD_y$ is greater than $MVD_x$, hence payload bit is extracted from the $MVD_y$ (instead of $MVD_x$). Next, the payload bits are extracted based on the predefined mapping rule for the merge mode candidates (see Fig. 2). Similarly, the embedded data can be extracted from the identified PBs by checking the block size based on the agreed-upon mapping rules, such as the one illustrated in Fig. 1.

## 5 Experimental results

The SHVC reference software SHM-12.0 [18] is modified to implement the proposed data embedding method. Experiments are conducted by using two-layer spatial scalability for Low Delay P (LDP) settings with group of pictures (GOP) structure of 4, i.e., IPPPIPPP$\cdots$. The scalable configuration with fixed quantization parameter (QP) is considered, where QP = 22 and 20 are set for the BL and EL, respectively. The remaining parameters are set to the SHM default configuration. A pseudo-random number generator is employed to generate a sequence $R$ of 0's and 1's, which is then embedded by manipulating the PSEs determined by the proposed method. Six standard test video sequences for SHVC from [22, 26], namely *BasketballDrill, BlueSky, FourPeople, PartyScene, RaceHorses* and *RushHour* are considered to evaluate the performance of the proposed data embedding techniques. The test video sequences considered in the experiments are sufficiently diverse, i.e., having complex textures and smooth regions of varying sizes and motions at various speeds. $R$ is embedded into the designated venues of the first 200 frames (which is the maximum length for one of the sequence - *BlueSky*, rounded to hundreds) in each test video sequence. It is verified that the processed video is still SHVC compliant, and the embedded data can be extracted correctly from the respective PSEs. By using SHVC compressed video as the baseline, the processed videos are evaluated in terms of variation in bit rate, video quality and payload. We conducted 9 sets of experiment using different combination of PSEs for data embedding. These combinations include: IPM, MVP, MVD, MVP+MVD (referred to as MVA), Merge Mode (MRG), Block size (BSZ), MVP+MVD+MRG (referred to as MVG), IPM+MVP+MVD+MRG (referred to as IVG), and IPM+MVP+MVD+MRG+BSZ (referred to as ALL).

### 5.1 Bit rate variation

The comparison of bit rate overhead among different combinations of PSEs is summarized in Table 1. Results indicate that, when multiple PSEs (e.g., IVG) are jointly utilized for data embedding, the results yield $+3.4\%$ of bit rate variation, which is relatively small in comparison to the achieved payload. One of the reasons is that, among the competing prediction modes, $RDO$ selects the one with the best cost for coding. In addition, the embedding only takes place at the block having the smallest size, i.e., $4 \times 4$ for intra-coded block, and $4 \times 8$ or $8 \times 4$ for inter-coded block. In comparison, Buhari et al.'s method [7] achieves an average bit rate overhead of $\sim 2.2\%$, which is slightly lower that ours, but it has lower a payload, although we acknowledge that their method was evaluated for H.264/SVC video. In general, manipulating the PB of different size causes higher bit rate overhead, while the

**Table 1** Comparison of bit rate overhead (%) for different data embedding techniques

| Test Sequence | IPM | MVP | MVD | MVA | MRG | MVG | IVG | BSZ | ALL |
|---|---|---|---|---|---|---|---|---|---|
| *RushHour* @30 Hz | 0.6 | 0.4 | 0.9 | 1.0 | 2.7 | 2.1 | 2.6 | 7.5 | 16.2 |
| *FourPeople* @60 Hz | 1.9 | 0.1 | 0.3 | 0.4 | 1.9 | 2.2 | 3.5 | 7.1 | 13.8 |
| *BlueSky* @24 Hz | 0.6 | 0.2 | 0.9 | 1.3 | 1.7 | 3.4 | 3.4 | 4.3 | 8.8 |
| *BasketballDrill* @50 Hz | 2.4 | 0.3 | 0.8 | 1.0 | 2.2 | 3.1 | 4.7 | 6.0 | 12.5 |
| *PartyScene* @50 Hz | 1.0 | 0.3 | 0.8 | 1.1 | 1.3 | 2.8 | 3.5 | 3.6 | 7.4 |
| *RaceHorses* @30 Hz | 0.5 | 0.6 | 0.9 | 1.0 | 0.9 | 2.4 | 2.6 | 3.4 | 6.5 |

**Table 2** Video quality degradation in term of PSNR (dB) for different data embedding techniques

| Test sequence | Resolution | IPM | MVP | MVD | MVA | MRG | MVG | IVG | BSZ | ALL |
|---|---|---|---|---|---|---|---|---|---|---|
| *RushHour* | BL (960 × 540) | 0.01 | 0.01 | 0.01 | 0.03 | 0.01 | 0.05 | 0.05 | 0.12 | 0.00 |
| | EL (1280 × 720) | 0.00 | 0.01 | 0.00 | 0.02 | 0.01 | 0.03 | 0.04 | 0.72 | 0.10 |
| *FourPeople* | BL (640 × 360) | 0.05 | 0.00 | 0.01 | 0.02 | 0.02 | 0.05 | 0.11 | 0.13 | 0.23 |
| | EL (1280 × 720) | 0.01 | 0.00 | 0.01 | 0.02 | 0.02 | 0.05 | 0.06 | 0.07 | 0.13 |
| *BlueSky* | BL (640 × 360) | 0.01 | 0.01 | 0.03 | 0.05 | 0.01 | 0.07 | 0.09 | 0.10 | 0.19 |
| | EL (1280 × 720) | 0.02 | 0.01 | 0.02 | 0.04 | 0.01 | 0.06 | 0.07 | 0.07 | 0.14 |
| *BasketballDrill* | BL (416 × 240) | 0.02 | 0.00 | 0.01 | 0.01 | 0.01 | 0.04 | 0.06 | 0.07 | 0.15 |
| | EL (832 × 480) | 0.01 | 0.01 | 0.01 | 0.02 | 0.01 | 0.04 | 0.05 | 0.06 | 0.12 |
| *PartyScene* | BL (416 × 240) | 0.03 | 0.01 | 0.02 | 0.03 | 0.00 | 0.05 | 0.09 | 0.07 | 0.15 |
| | EL (832 × 480) | 0.03 | 0.00 | 0.01 | 0.02 | 0.01 | 0.03 | 0.07 | 0.05 | 0.11 |
| *RaceHorses* | BL (416 × 240) | 0.02 | 0.07 | 0.08 | 0.09 | 0.06 | 0.10 | 0.07 | 0.14 | 0.15 |
| | EL (832 × 480) | 0.01 | 0.04 | 0.04 | 0.04 | 0.04 | 0.05 | 0.02 | 0.08 | 0.06 |

**Table 3** Payload (kbps) for different data embedding techniques

| Sequence | Resolution | IPM | MVP | MVD | MVA | MRG | MVG | IVG | BSZ | ALL |
|---|---|---|---|---|---|---|---|---|---|---|
| *RushHour* | BL | 7.8 | 8.7 | 8.3 | 13.0 | 18.1 | 38.0 | 45.7 | 58.4 | 126.7 |
| | EL | 5.4 | 3.4 | 3.1 | 6.8 | 21.0 | 46.8 | 52.0 | 70.5 | 162.5 |
| | Total | 13.1 | 12.1 | 11.4 | 19.8 | 39.1 | 84.8 | 97.7 | 128.9 | 289.2 |
| *FourPeople* | BL | 39.2 | 4.4 | 4.0 | 7.0 | 19.9 | 35.5 | 75.1 | 32.5 | 173.5 |
| | EL | 63.2 | 10.3 | 9.2 | 14.6 | 64.0 | 101.3 | 174.0 | 112.8 | 458.0 |
| | Total | 102.5 | 14.7 | 13.1 | 21.6 | 83.8 | 136.7 | 249.2 | 145.4 | 631.5 |
| *BlueSky* | BL | 9.4 | 7.8 | 6.6 | 10.6 | 11.5 | 34.9 | 43.1 | 34.5 | 76.6 |
| | EL | 33.7 | 26.0 | 22.0 | 35.6 | 43.5 | 116.3 | 146.0 | 115.3 | 257.7 |
| | Total | 43.1 | 33.9 | 28.6 | 46.2 | 54.9 | 151.3 | 189.1 | 149.8 | 334.3 |
| *BasketballDrill* | BL | 18.9 | 6.3 | 5.6 | 10.5 | 11.8 | 29.3 | 48.4 | 44.1 | 96.1 |
| | EL | 63.1 | 16.2 | 14.3 | 25.4 | 35.2 | 84.5 | 149.4 | 178.5 | 292.7 |
| | Total | 82.0 | 22.5 | 19.9 | 36.0 | 47.0 | 113.8 | 197.8 | 222.6 | 388.8 |
| *PartyScene* | BL | 30.8 | 11.0 | 9.5 | 16.9 | 17.1 | 48.6 | 78.8 | 47.0 | 114.5 |
| | EL | 112.4 | 45.5 | 38.2 | 64.3 | 66.8 | 195.8 | 299.4 | 64.9 | 441.6 |
| | Total | 143.2 | 56.5 | 47.8 | 81.2 | 83.9 | 244.4 | 378.2 | 112.0 | 556.1 |
| *RaceHorses* | BL | 13.2 | 7.6 | 7.2 | 12.6 | 7.5 | 25.2 | 30.6 | 37.2 | 65.9 |
| | EL | 30.8 | 17.9 | 17.2 | 28.4 | 28.1 | 66.1 | 77.1 | 103.2 | 163.5 |
| | Total | 44.0 | 25.4 | 24.4 | 41.0 | 35.6 | 91.3 | 107.7 | 140.4 | 229.4 |

BSZ technique results in an average bit rate overhead of 5.3%. One of the reasons is that manipulating larger block requires more bits for coding the prediction residual. When BSZ is jointly utilized with IVG, the average bit rate overhead increases by 10.9%.

### 5.2 Video quality

The comparison of quality degradation in term of PSNR (dB) between the original (compressed) and processed (compressed+payload) videos are recorded in Table 2. It is observed that, in most cases, the manipulation of PSEs leads to insignificant quality degradation. Specifically, the average degradation in PSNR ranges from 0.02 dB to 0.11 dB for IVG. In the worst case scenario, PSNR drops by 0.72 dB for the sequence *RushHour* for BSZ. On the other hand, in the best case scenario, when using MVP, the drop in PSNR is $< 0.07$ dB, where the average drop in PSNR is $\sim 0.01$ dB. Overall result suggests that the manipulation of PSEs leads to degradation in video quality. In comparison to IVG, the average PSNR degradation observed in Buhari et al's method [7] falls in the ranges of [0.04, 0.36] dB.

### 5.3 Payload

Table 3 records the payload for different combinations of PSEs. As expected, the payload consistently increases when more PSEs are utilized for data embedding purposes. Encouraging results are achieved by ALL, which is evidenced in the Table 3. The combined use of PSEs has the advantage of being able to choose the PSE having $RDC_{best}$ to represent the payload bits. However, to achieve optimal PSE, the RDO needs to check the $RDC$ for every possible prediction mode, which increases the complexity of the encoder.

While suppressing bit rate overhead and quality degradation, the attained payload in ALL is significantly more ($\sim 3\times$) in comparison to existing solutions which only utilize one single PSE. For example, in the sequence *PartyScene*, the achieved payload when using ALL is $3.9\times$ more than that of IPM.

### 5.4 Impact to coding structure and prediction mode

The impact of CUPSEED for data embedding on the coding block structure and prediction mode are analyzed. To facilitate the discussion, a short description of each test video sequence is provided in Appendix to highlight its characteristics.

Figure 6 shows the number of blocks coded at different partitioning depth level when different PSEs are utilized for data embedding. Here, ORI refers to the statistics of the original video, i.e., compressed video without embedded data. Let $D_4$ represent the splitting depth for $4 \times 4$ block. It is observed that the sequence *RushHour, BlueSky* and *RaceHorses* are coded with more larger blocks, while the other sequences have more $4 \times 4$ blocks coded. Usually, after attempts are made to embed payload bit into a particular block at partitioning depth $D_{i+1}$, the total $RDC_{D_{i+1}}$ for the block is increased. This leads to the situation where the total $RDC_{D_{i+1}}$ being larger than that of $RDC_{D_i}$ (i.e., parent block). For example, a $2N \times 2N$ PU (parent) at partitioning depth level $D_i$ is split into four $N \times N$ PUs (children) (i.e., at partitioning depth $D_{i+1}$) for intra and/or inter picture prediction. After the $N \times N$ PUs are manipulated to embed data, the total $RDC$ for these split blocks are increased. If the total $RDC_{D_{i+1}}$ for the split blocks are greater than the parent block, then the $2N \times 2N$ PU is selected for coding instead of the four-split PUs. In most data embedding techniques (except BSZ and its combination), the distribution of blocks is shifted to the larger blocks (or smaller blocks if available) after data embedding. In general, it is observed that the number of $4 \times 4$

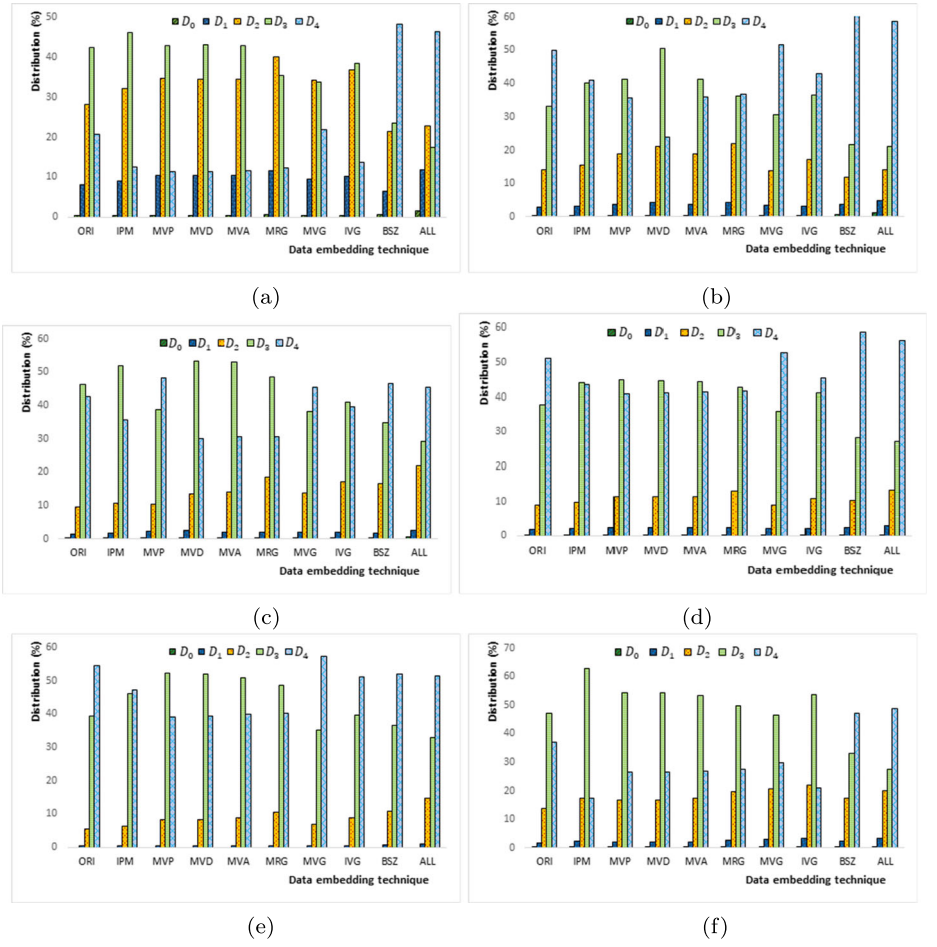


**Fig. 6** Distribution of coding blocks at different partitioning depth level for different data embedding techniques. **a** RushHour **b** FourPeople **c** BlueSky **d** BasketballDrill **e** PartyScene **f** RaceHorses

blocks for IPM increase when BSZ is used for the sequences which originally contain more smooth regions such as the sequences *RushHour*, *FourPeople* and *RaceHorses*. One of the reasons is that $P(m_i = 0) \sim 0.5$, which causes the mode-split distribution to change after embedding data by using the mapping rules between $m_i$ and PSE.

Figure 7 shows the distribution of prediction modes for the original and processed videos. It is observed that there are more merge mode being coded for the sequences *RushHour* and *BlueSky* in the original video. This is because the sequence *RushHour* contains non-static background and the sequence *BlueSky* contains moving background. The sequences *FourPeople*, *BasketballDrill* and *PartyScene* contain either more static objects/background or fine textures while the sequence *RaceHorses* contains relatively more large smooth regions. Hence, these videos are originally coded with more IPM. When embedding data by using IPM, the distribution of prediction modes shifted to more MV and merge mode,

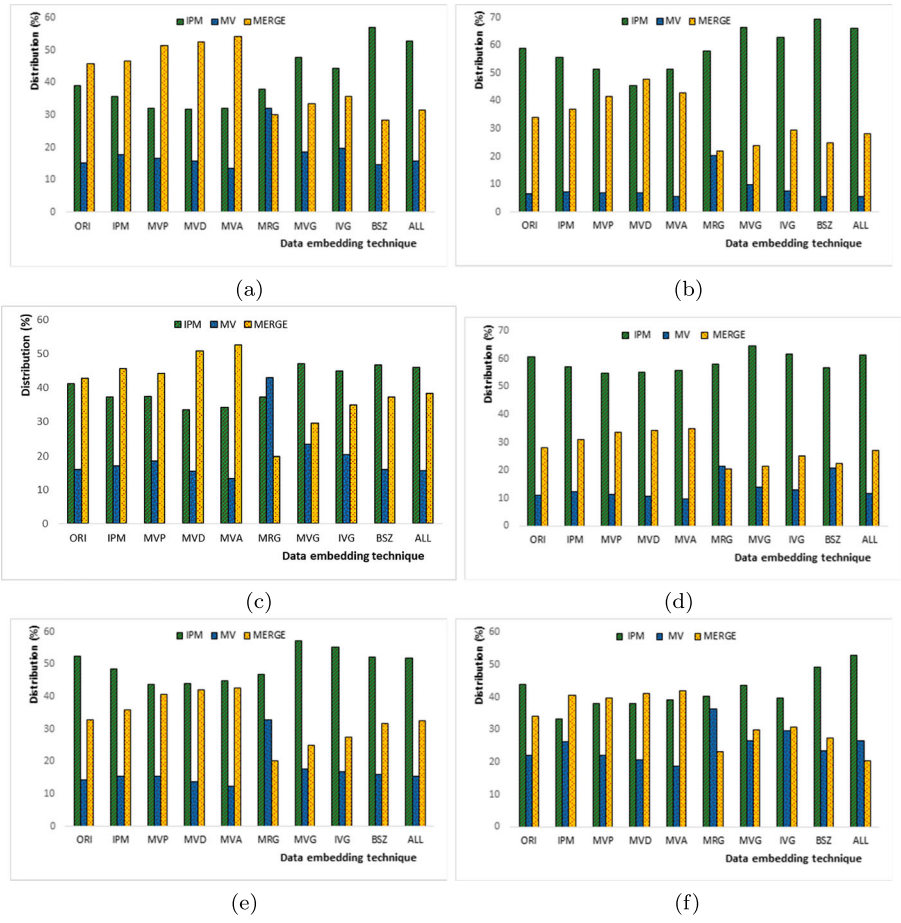


**Fig. 7** Distribution of prediction modes for different data embedding techniques. **a** RushHour **b** FourPeople **c** BlueSky **d** BasketballDrill **e** PartyScene **f** RaceHorses

and in some cases blocks with larger size. When MVs are utilized for data embedding (i.e., MVP and MVD), the resulting video tends to have more merge modes. There are more MVs coded when MRG is utilized for data embedding. The trend is slightly different for BSZ, where more $4 \times 4$ blocks are coded. It can be concluded that when a particular prediction mode is utilized for data embedding, it results in higher $RDC$, and hence either other prediction mode or other block size with better $RDC$ is selected for coding.

Recalled that two levels of decision are made by RDO: block size and prediction mode. These results demonstrate that manipulating PSEs affects the distribution of coded block size and prediction mode. After data embedding, video sequence with more static background tends to be coded by using IPM with different block size or merge mode. This trend can be observed in the sequence *FourPeople*. When MV is manupulated, more blocks are coded in merge mode, which can be observed in the sequence *RushHour* and *BlueSky*. However, when merge mode is manipulated, more IPM is coded, which can be observed in the sequence *BasketballDrill* and *PartyScene*.

### 5.5 Discussion

IPM, MVP and MVD are exploited to embed data by manipulating the parity bit of the PSEs, while MRG and BSZ associate the respective PSEs with payload bit. Among these five venues, BSZ achieves the highest payload, followed by IPM and MRG, depending on the video content. Specifically, video with more motion (e.g., *RushHour*) favors MRG. On the other hand, video sequence with more static object / background (e.g., *FourPeople*) favors IPM. Subsequently, MVP and MVD also offer some venues to embed data.

In general, the availability of prediction mode determines the size of the payload. Specifically, BSZ can be applied to the I, P and B-slices, hence more blocks are available for manipulation. On the other hand, IPM is only applicable to I-slices, which restricts the payload. While MVP, MVD and MRG are applicable to the P and B-slices, PSEs involves MV, which is relatively low in existence.

It is noteworthy that embedding data into all venues (PSEs) causes minimal impact on video quality. This is because after data embedding attempts are made to IPM, MV (i.e., MVP and MVD) and MRG, only the one with $RDC_{best}$ is coded as illustrated in Algorithm 3. The result is more encouraging as compared to using single venue embedding because the selection of PSEs to associate with the data bit is the best among all PSEs. Moreover, after each venue is utilized for data embedding, the prediction error/residual is recomputed for reconstruction purposes. The advantage of utilizing these PSEs is that propagation of error can be kept to the minimum in comparison to the manipulation of the transformed-quantized coefficients. While transformed-quantized coefficients can provide significantly higher payload, taking such approach may lead to noticeable quality degradation and error propagation when more coefficients are exploited for embedding.

The bit rate overhead for using MVP, MRG and MVD are relatively low in comparison to the payload. However, the bit rate overhead for using BSZ is higher due to PBs of varying block sizes being manipulated. In general, altering bigger block introduces higher prediction error and more bits are required to code the residual. These techniques maintain coding efficiency with some increase in bit stream size. Hence, a straight forward approach for improving bit rate is to restrict data embedding to smaller blocks.

CUPSEED increases the complexity of the video encoding process, but it is an encouraging solution for achieving high payload and high video quality while suppressing bit rate overhead simultaneously. Results also suggest that when all PSEs are utilized, they can collectively host $\sim 3\times$ more payload in comparison to the conventional single-venue data embedding technique.

### 5.6 Recommendations

With the aforementioned results, the following recommendations are put forward when utilizing PSEs for data embedding:

(a) In general, manipulating blocks of smaller block sizes has less impact to the bit rate variation and video quality. Hence, it is suggested that data embedding in IPM is feasible for most cases, especially for smaller GOP sizes where there are more I-slice. In addition, IPM can be applied to all coded layers since embedding in the BL and EL layers using opposite directions can suppress the bit rate variation and preserve the video quality in ELs. The achievable payload for video sequence with more static object/background (e.g., *FourPeople*) is more encouraging. Besides, the video with more fine regions or small objects (e.g., *PartyScene*) can achieve higher payload. All in all, IPM is found to be the most promising individual PSE for data embedding, where

an average of 71.3 kbps can be embedded into the $4 \times 4$ blocks with bit rate increment of 1.2% and quality degradation of 0.02 dB.

(b) The payload for video sequences with more motion (e.g., *RushHour* and *BlueSky*) is more encouraging when using MVP and MVD. A combined used of MVP and MVD increases the payload, while $< 2\%$ bit rate overhead and slight quality degradation are observed. Therefore, MVP and MVD are suggested for the sequence with more motion. When ILR picture is used as a reference picture for predicting EL (i.e., MV in EL set to zero), these PSEs can only be applied to BL. Here, IPM, MVD and MVP only utilize the smallest blocks for data embedding. However, IPM, MVD and MVP can be extended to large block size to cater for more payload. In contrast, for applications that require less payload, these PSEs provide the flexibility of selecting blocks at different partitioning depth level and prediction mode to embed the payload.

(c) Merge mode was introduced in HEVC standard to improve the compression efficiency irregardless of the motion speed. The experimental results show that there are more merge mode coded for the sequences with more motion, which implies that higher capacity can be achieved by using the MRG technique. Therefore, MRG is also suggested for the sequence with more motion. Here, MRG is applied to all coded layers and the experimental results demonstrate that it can achieve encouraging payload. For instance, the sequence *BlueSky* and *RushHour* offer higher payload in comparison to IPM, with bit rate variation of $< 3\%$ and video degradation $\leq 0.06$ dB. Regardless of the video content, MRG outperforms MV-based techniques as suggested by the results recorded in Table 3. In addition, the MVG and IVG techniques offers higher payload as observed in Table 3. With IVG, the payload is $2\times$ more of that of single PSE, while the bit rate overhead is $< 5\%$. Hence, it is recommended to use MVG or IVG for applications requiring higher payload. In contrast, when the demand of payload is not high, this combination provides the flexibility to select blocks at different partitioning depth level for data embedding.

(d) The quad-tree block partitioning structure provides greater flexibility in manipulating the block size in a video. BSZ can be applied to smaller blocks only if bit rate overhead is a concern. To further suppress bit rate variation, BSZ can be applied to a single layer. Here, blocks for all partitioning depth level are manipulated for data embedding purpose. Fig. 6 and 7 suggest that more IPMs are coded when BSZ is applied. Therefore a combination of BSZ and IPM can also be applied to increase the payload.

## 6 Conclusions

A combined use of prediction syntax elements, including IPM, MVP, MVD, MRG and BSZ, is put forward to embed data in SHVC coded video. Experiments results suggest that the proposed data embedding method, called CUPSEED, outperforms single-venue PSE in terms of payload. Results suggest that when all PSEs are utilized, they can collectively host $3\times$ more payload in comparison to the conventional data embedding technique. In the best case scenario, 556.1 kbps payload is embedded into the sequence *PartyScene* with a drop of 0.15 dB in PSNR and an overhead of 7.4% in bit rate. The analysis on the data embedding impact to the prediction modes as well as coding structure of SHVC video are conducted. It is observed that when embedding data into the smallest block, either more larger blocks or other prediction mode, whichever has the better rate-distortion-cost, is coded. Specifically, the texture and motion data decides the proportion of prediction modes and block size.

For future work, we will extend the proposed method for actual application in video management, annotation, hyper-linking and etc. We also intent to extend the combined use of prediction syntax elements together with other syntax elements such as transformed coefficient and quantization parameter for data embedding purposes.

**Acknowledgments** This work was supported by the Advanced Engineering Platform's Cluster Funding (account number AEP-2020-Cluster-04), Monash University Malaysia, Malaysia.

## Appendix

**RushHour** A sequence showing a street in a city with heavy traffic. There are many cars captured at various distances and displayed as smooth regions of different sizes. The camera stands still and most motions are related to the moving cars. The background is not completely static due to the air ripples caused by heat.

**FourPeople** This is a scene with four people sitting and passing brochure from one to another. There are areas rich in textures such as the bunting in the background, but also some smoother areas such as the wall and tables. The camera is stationed in a fixed position. The background is also static. Most motions are related to the hands of the four people for passing brochure.

**BlueSky** A very high contrast sequence showing dark leaves of a tree with bright blue sky and relatively smooth rotating motion. Some areas are very smooth (blue sky) while some areas are very complex with high contrast borders between the tree and the sky.

**BasketballDrill** This is a sequence with much motion, where several basketball players are performing shooting drills, running and turning quickly while the camera fixed and the background is static. The background is relatively rich in details and textures.

**PartyScene** A scene with few children, a pile of colorful Christmas gifts, a complex brick background, and a Christmas tree on the left. Two children are running around a tree on the right hand side, a girl is blowing bubbles, while a duck toy is dancing to the left and right. This sequence has small objects with complex motion and much fine texture across the whole frame. The camera pans to the direction of the girl who is blowing bubbles.

**RaceHorses** This is a sequence with few people riding on horses and moving around. The camera is panning to focus on the moving object. The background is not static and the foreground has a mixture of smooth and detail region.

## References

1. Abdulla AA, Sellahewa H, Jassim SA (2014) Steganography based on pixel intensity value decomposition. In: Agaian SS, Jassim SA, Du EY (eds) International Society for Optics and Photonics Mobile Multimedia/Image Processing, Security, and Applications 2014, vol 9120. SPIE, pp 19–27
2. Abdulla AA (2015) Exploiting similarities between secret and cover images for improved embedding efficiency and security in digital steganography. University of Buckingham
3. Abdulla AA, Sellahewa H, Jassim SA (2019) Improving embedding efficiency for digital steganography by exploiting similarities between secret and cover images. Multimed Tools Appl 78:17799–17823

4. Aly HA (2011) Data hiding in motion vectors of compressed video based on their associated prediction error. IEEE Trans Inf Forensic Secur 6(1):14–18
5. Boyce JM, Ye Y, Chen J, Ramasubramonian AK (2016) Overview of SHVC: Scalable extensions of the high efficiency video coding standard. IEEE Trans Circ Syst Video Technol 26(1):20–34
6. Bross B, Helle P, Lakshman H, Ugur K (2014) High efficiency video coding (HEVC) algorithms and architectures. Springer, Cham
7. Buhari AM, Ling H-C, Baskaran VM, Wong K (2016) Fast watermarking scheme for real-time spatial scalable video coding. Signal Process Image Commun 47:86–95
8. Fang D-Y, Chang L-W (2006) Data hiding for digital video with phase of motion vector. In: IEEE International Symposium on Circuits and Systems, pp 1422–1425
9. Guan Z, Wu HT (2020) A reversible contrast enhancement scheme for color images. In: 2020 IEEE International Conference on Multimedia and Expo (ICME), pp 1–6
10. Konyar MZ, Akbulut O, ztrk S (2020) Matrix encoding-based high-capacity and high-fidelity reversible data hiding in HEVC. Signal Image Video Process 14:897–905
11. Nguyen C, Tay DBH, Deng G (2006) A fast watermarking system for h.264/avc video. In: APCCAS 2006 - 2006 IEEE Asia Pacific Conference on Circuits and Systems, pp 81–84
12. Ohm J, Sullivan GJ, Schwarz H, Tan TK, Wiegand T (2012) Comparison of the coding efficiency of video coding standardsincluding high efficiency video coding (HEVC). IEEE Trans Circ Syst Video Technol 22(12):1669–1684
13. Pang L, Wong K, Liong ST (2017) Data embedding in scalable coded video. In: 2017 asia-pacific signal and information processing association annual summit and conference (apsipa), pp 1190–1194
14. Pang L, Wong K (2019) A data embedding technique for spatial scalable coded video using motion vector predictor. In: 2019 IEEE International Conference on Image Processing (ICIP), pp 4050–4054
15. Pang L, Wong K, Ito R (2019) Merge mode-based data embedding in SHVC compressed video. In: 2019 International Symposium on Intelligent Signal Processing and Communication Systems (ISPACS), pp 1–2
16. Shanableh T (2018) Altering split decisions of coding units for message embedding in HEVC. Multimed Tools Appl 77(7):8939–8953
17. Sheng Q, Wang R, Pei A, Wang B (2016) An information hiding algorithm for HEVC based on differences of intra prediction modes. In: Sun X, Liu A, Chao H-C, Bertino E (eds) Cloud Computing and Security. Springer International Publishing, Cham, pp 63–74
18. SHM (2017) https://hevc.hhi.fraunhofer.de/svn/svn_SHVCSoftware/
19. Tew Y, Wong K (2014) Information hiding in HEVC standard using adaptive coding block size decision. In: 2014 IEEE International Conference on Image Processing (ICIP), pp 5502–5506
20. Tew Y, Wong K (2014) An overview of information hiding in H.264/AVC compressed video. IEEE Trans Circ Syst Video Technol 24(2):305–319
21. Tew Y, Wong K, Phan RC-W, Ngan KN (2018) Separable authentication in encrypted HEVC video. Multimed Tools Appl 77(18):24165–24184
22. Universität-Hannover (2013) Test sequence. ftp://ftp.tnt.uni-hannover.de/testsequences/
23. Van LP, Praeter JD, Wallendael GV, Cock JD, de Walle RV (2015) Out-of-the-loop information hiding for HEVC video. In: 2015 IEEE International Conference on Image Processing (ICIP), pp 3610–3614
24. Wang J, Wang R, Xu D, Li W (2015) An information hiding algorithm for HEVC based on angle differences of intra prediction mode. JSW 10(2):213–221
25. Wong K, Chan C, MaungMaung A (2020) Lightweight authentication for MP4 format container using subtitle track. IEICE Trans Inf Syst E103.D(1):2–10
26. Xiph (2013) Derf's collection. https://media.xiph.org/video/derf
27. Xu J, Wang RD, Huang ML, Wang JJ, Xu DW (2015) An information hiding algorithm for HEVC based on intra-prediction modes and hamming+ 1. J Comput Inf Syst 11(15):5587–5598
28. Xu C, Ping X, Zhang T (2006) Steganography in compressed video stream. In: First International Conference on Innovative Computing, Information and Control - Volume I (ICICIC'06), vol 1, pp 269–272
29. Yang J, Li S (2018) An efficient information hiding method based on motion vector space encoding for HEVC. Multimed Tools Appl 77(10):11979–12001
30. Yang Y, Li Z, Xie W, Zhang Z (2019) High capacity and multilevel information hiding algorithm based on pu partition modes for HEVC videos. Multimed Tools Appl 78(7):8423–8446

## Affiliations

**LieLin Pang[1] · Yiqi Tew[2] · KokSheik Wong[3,4] · Mohamad Nizam Bin Ayub[1]**

LieLin Pang
adpangll@siswa.um.edu.my

Yiqi Tew
yiqi@tarc.edu.my

Mohamad Nizam Bin Ayub
nizam_ayub@um.edu.my

1 University of Malaya, Kuala Lumpur, Malaysia

2 Tunku Abdul Rahman University College, Kuala Lumpur, Malaysia

3 School of Information Technology, Monash University, Selangor, Malaysia

4 Advanced Engineering Platform, Monash University, Selangor, Malaysia