Shanableh, T. Multimed Tools Appl (2017). doi:10.1007/s11042-017-4787-6
The final publication is available at Springer via http://dx.doi.org/10.1007/s11042-017-4787-6

Altering Split Decisions of Coding Units for Message Embedding in HEVC

Tamer Shanableh

Abstract This paper proposes a novel message embedding solution based on modifying the split decisions of HEVC videos. The encoder starts by computing a mapping between the split decisions of a Coding Unit (CU) and its features variables. This results in model weights that can be used to predict the split decisions. Message embedding is then carried out as a function of the predicted and true split decisions per CU. If the message bit to embed is '1' and the predicted and the true split decisions are different then the true split decision is modified to be identical to the predicted flag. Otherwise if the message bit is '0' and the predicted and the true split decisions are identical, then the true split flag is modified to become different than the predicted flag. We apply this embedding concept at two CU coding levels; 32x32 and 16x16. To extract a message, the decoder starts by regenerating the model weights which are then used to predict split decisions and compare them against the true decisions received in the bit stream. Identical decisions indicate a message bit of '1' and non-identical split decisions indicate a message bit of '0'. In the experimental results we examine the prediction accuracy, the effect of the proposed solutions on message payload, the number of modified split decisions and the corresponding impact on video quality. Comparison with an existing solution reveals that the proposed solution is superior in terms in message payload whilst resulting in reduced video distortions.

Keywords Digital video coding; HEVC; Data hiding; Pattern recognition.

1 Introduction

Message embedding in coded video is used as means of information hiding and has a number of important applications such as copyright protection [1], access control [2], content authentication [3] transaction tracking [4], real-time video scene change detection [5] and error detection and concealment [6]. In all applications, message embedding has a number of requirements including minimal video quality degradation, maintaining compatibility with the standardized bit stream syntax and offering a reasonable message payload.

Message embedding in compressed videos include modifying DCT coefficients, quantization scales and motion vectors. Message embedding can also be implemented using code-word substitution [7]. Matrix encoding can be used to modify the quantization scales and motion vectors in signal layer and scalable video coding [8]. Advanced transcoding techniques were applied to embed messages is pre-encoded video as well. The work in [9] embeds messages through altering the quantization scales using a machine learning approach. Data embedding using MVs has also been used for video watermarking [10]. Additionally, [11] proposed a solution that improves the security of motion vector-based data embedding. On the other hand, since embedding message by modulating MVs become a popular approach, detection of such approaches became an important research topic as reported in [12] and [13].

Message embedding is also implemented using coding block structure and prediction modes. For instance, the work in [14] proposed the alteration of intra prediction modes to hide a message in 4x4 intra blocks in H264/AVC videos. Likewise, the authors of [15] altered the block types of H.264/AVC blocks to hide message bits.

Message embedding is also implemented in HEVC videos in which messages are hid by forcing certain partitioning types for the Prediction Units (PUs) [16]. If the message bit to hide is '1'then the PU type is restricted to: 2Nx2N, nRx2N, Nx2N and 2NxnD. Whereas if a message bit is '0' then the PU type is restricted to: NxN, nLx2N, 2NxnU and 2NxN. The work in [17] proposed to embed messages into HEVC videos by modifying the 4x4 intra prediction modes. The solution has low impact on video quality, however because it is restricted to 4x4 intra prediction modes, it can only embed low payload

Shanableh, T. Multimed Tools Appl (2017). doi:10.1007/s11042-017-4787-6
The final publication is available at Springer via http://dx.doi.org/10.1007/s11042-017-4787-6

messages. The authors reported a message payload in the range of 2.2 to 7 Kbit/s.

In this work, we propose to embed messages in HEVC video by modifying the split decisions of either 32x32 or 16x16 CUs. Since each 64x64 CU contains four 32x32 CUs and sixteen 16x16 CUs, then 4 and 16 message bits can be hidden in the syntax of a CU respectively. A prediction solution is proposed in which the decoder can predict the split decisions of a CU and compare it against the true split decisions received in the video bit stream. If the two decisions are identical then the message bit is '1' otherwise it is '0'.

The challenge presented in is this work is to embed a message by modifying selective CU split decisions. The decoder should be able to identify the CU split decisions that curry message information and extract the message bits correctly. At the same time, it is desired to minimize the effect of message embedding on video quality. It is also desired to reduce the number of modified split decisions. All of these challenges are addressed in the proposed solutions of this paper.

We study the effect of such solutions on the amount of split decisions that need to be modified. We also examine the effect on video quality and the amount of message bits that can be embedded per a second of video.

The rest of this paper is organized as follows. Section 2 introduces the idea behind the overall message embedding system including the operations of the encoder and the decoder. Section 3 introduces the proposed feature extraction and model generation solutions. Section 4 presents the experimental results and Section 5 concludes the paper.

2. Proposed system

The main idea of the proposed system is to modify the split decisions of a 64x64 CU according to the message to be embedded. It is known that a 64x64 CU has a total of 21 split flags. These are allocated as follows; five split flags for each 32x32 CU and one split flag for the 64x64 CU. Each 32x32 CU has four 16x16 split flags and one 32x32 split flag. The proposed data embedding system hides message bits in the split decisions of 16x16 CUs. Hence a total of 16 flags can be manipulated per 64x64 CU.

The proposed system is divided into two subsystems; message embedding and message extraction.

The message embedding subsystem is composed of two stages. In the first stage, modeling takes place to compute a mapping between coded information of surrounding CUs and the split flags of the current CU. This results in model weights that can be used for predicting the split decisions. Typically, the first 10% of a video sequence is used to generate the model. However the model generation can be repeated if needed. For instance, in [18] it was suggested to repeat model generation in cases of scene cut detection. The overall model generation process is illustrated in Figure 1.

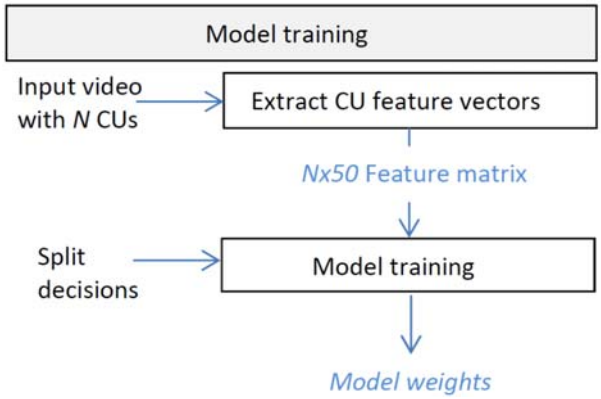

Figure 1. Overall model generation process

In the figure, 50 feature variables are used to compose a feature vector for each CU. These feature variables are collected from surrounding CUs that are previously encoded. Namely, the following CUs are used in the feature collection process; the one on the left, top-left, top, top-right and co-located CUs. The details of feature collection and model generation are explained in the next section. It is important to note that collected feature variables are available for both the encoder and the decoder, as the latter will repeat the same process to compute the model weights.

The second stage of the message embedding subsystem is the embedding of message bits. This starts once the encoder computed the model weights. We propose two solutions for message embedding. In the first solution, a message is embedded by modifying the 16x16 split decisions of a 64x64 CU. In the second solution, a message is embedded by modifying the 32x32 split decisions of a 64x64 CU.

2.1 Message embedding by modifying 16x16 split decisions.

Shanableh, T. Multimed Tools Appl (2017). doi:10.1007/s11042-017-4787-6
The final publication is available at Springer via http://dx.doi.org/10.1007/s11042-017-4787-6

In this solution, three pieces of information are need as follows; a message bit to embed, predicted split flag using the model and the true split flag computed by the encoder. If the message bit to embed is '1' then the encoder uses a split flag that is identical to the predicted one. Else, if the message bit to embed is '0' then the encoder uses a split flag that is different than the predicted one. This process of data embedding is illustrated in Figure 2.

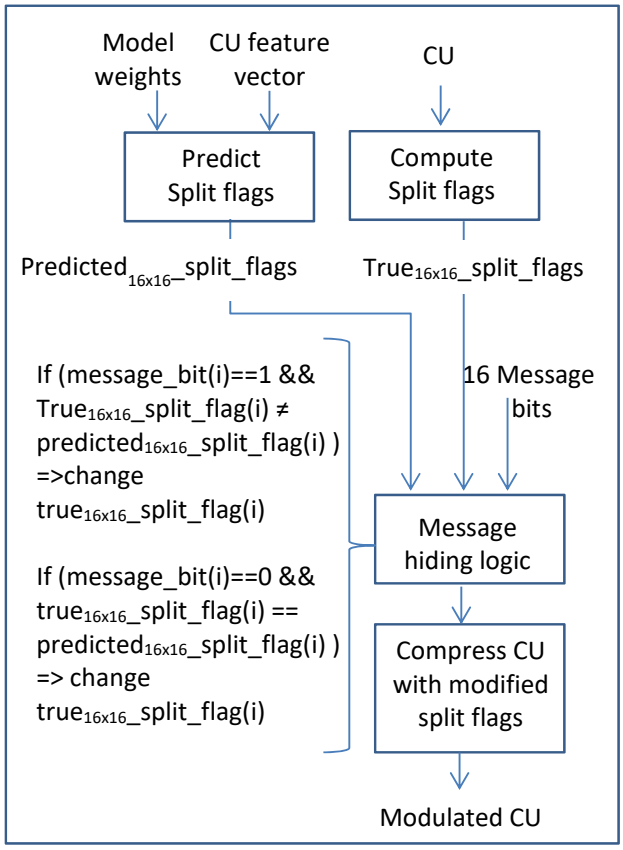

Figure 2. Illustration of message embedding process.

As mentioned previously, the proposed system is divided into two subsystems; message embedding and message extraction. In the message extraction subsystem, the decoder starts by re-computing the model. This is why it is important to ensure that all feature variables are available for both the encoder and the decoder. Thereafter, for the decoder to extract a message bit, 2 pieces of information are needed as follows; a true split flag decoded from the bit stream and a predicted split flag computed using the model. If the two flags are identical then the message bit is '1' else it is '0'. This process of message extraction is illustrated in Figure 3.

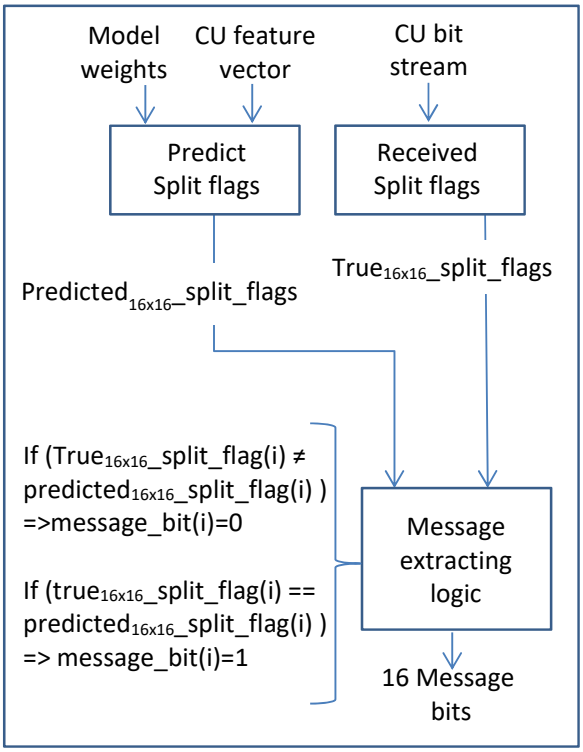

Figure 3. Illustration of message extraction process.

2.2 Message embedding by modifying 32x32 split decisions.

In this solution, the split decisions of 64x64 CUs are predicted using the model weights. If the message bit to embed is '1' then the encoder is forced to use the predicted split pattern, otherwise if the bit is '0', the encoder selects the split pattern using the usual rate-distortion optimization approach. This results in embedding one message bit per 64x64 CU., the embedding concept is used at the level of 32x32 CUs. The split decisions of 64x64 CUs are predicted which contain the split decisions of four 32x32 CUs. If a message bit is '1' then the split decisions for a 32x32 CU are replaced by the predicted pattern. Otherwise, the encoder selects the split pattern using the usual rate-distortion optimization approach. However, in this case, if the predicted and encoder-selected split decisions are the same, the encoder is required to modify one of the split decisions of the 32x32 CU.

As such, up to 4 bits can be embedded in each 64x64 CU. The message embedding algorithm is further explained in the pseudocode of Figure 4.

Shanableh, T. Multimed Tools Appl (2017). doi:10.1007/s11042-017-4787-6
The final publication is available at Springer via http://dx.doi.org/10.1007/s11042-017-4787-6

```
-------------------------------------------------------------------
ALGORITHM embedMessage is
-------------------------------------------------------------------
INPUT:   videoToEncode,
         trainFVs,
         trainSplitFlags
OUTPUT:  videoWithEmbeddedMsg

H = computePredictionModel(trainFVs, trainSplitFlags)
REPEAT
  INPUT currCU;
  skipFlag = computeSkipFlag(currCU);
  if !skipFlag
    currFV = computeFV(currCU);
    predictedSplit = predictSplit(currFV, H);
    actualSplit = computerSplit(currCU);
    INITIALIZE modulatedSplit = zeros(1,21);
    INITIALIZE i=0; a=1; b=5;
    messageBits = readNext4MsgBits();
    REPEAT
      if actualSplit(a:b) == predSplit(a:b)
        actualSplit(a:b) =
              alterOneBitInSplit(actualSplit(a:b));
      end
      if messageBits(i)==1
          modulatedSplit(a:b) = acutalSplit(a:b)
      else
          modulatedSplit(a:b) = predictedSplit(a:b)
      end
      i+=1; a+=5; b+=5;
      UNTIL 4 message bits are embedded
      videoWithEmbededMsg =
              concatinate(videoWithEmbededMsg,
                    encodeCU(currCU, modulatedSplit);
  end //if !skipFlag
```

Figure 4. Pseudocode for the message embedding algorithm.

An example that illustrates this concept in shown in Figure 4. In the figure, the split decisions/flags of each 32x32 CU are composed of four 16x16 CU split flags followed by the split flag of the entire 32x32 CU.

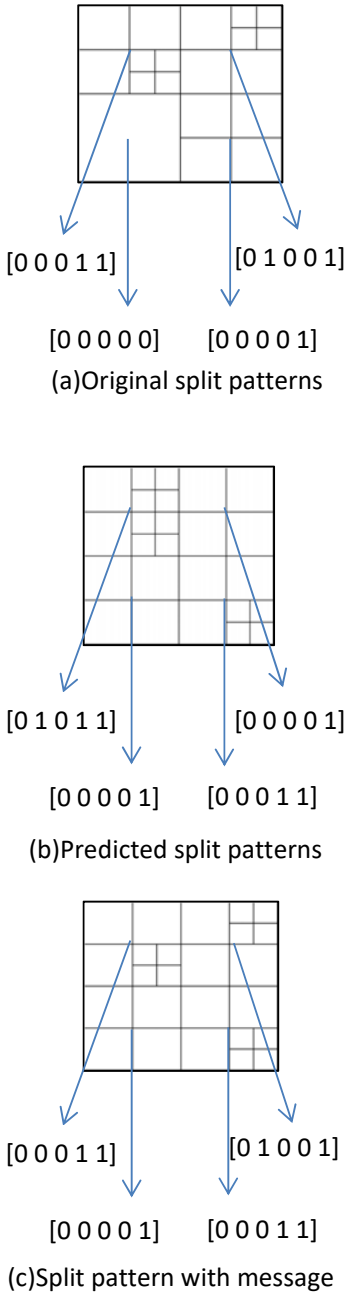

Figure 3. Example of embedding [0 0 1 1] into the split patterns of four 32x32 CUs

In the example, the original split decisions selected by the encoder is shown in part 'a'. The predicted split decisions on the other hand are shown in part 'b'. And the split decisions with data embedding are shown in part 'c'. The message to embed in this example is 0011, therefore, the split decisions of the last two 32x32 CUs in part 'a' are replaced with predicted split values of [0 0 0 0 1] and [0 0 0 1 1] from part 'b'. The final 64x64 CU split decisions

Shanableh, T. Multimed Tools Appl (2017). doi:10.1007/s11042-017-4787-6
The final publication is available at Springer via http://dx.doi.org/10.1007/s11042-017-4787-6

which contains the embedded message are therefore a mix between parts 'a' and 'b' as shown in part 'c'. To extract the embedded message at the decoder, for each 64x64 CU, a feature vector is composed from surrounding CUs. The model is then used to predict the split pattern. The predicted pattern is then compared against the actual split decisions received in the bit stream for each 32x32 CU. If they match then the embedded message bit is '1', otherwise it is '0'. The message extraction algorithm is further explained in the pseudocode of Figure 5.

```
-------------------------------------------------------------------------
ALGORITHM extractMessage is
-------------------------------------------------------------------------
INPUT:   videoBitstream,
         trainFVs,
         trainSplitFlags
OUTPUT: embeddedMessage

H = computePredictionModel(trainFVs, trainSplitFlags)
REPEAT
  currCU = decodeNextCU(videoBitstream);
  skipFlag = readSkipFlag(currCU);
  if !skipFlag
    currFV = computeFV(currCU, videoBitstream);
    predictedSplit = predictSplit(currFV, H);
    actualSplit = computerSplit(currCU);
    INITIALIZE messageBits = zeros(1,4);
    INITIALIZE i=0; a=1; b=5;
    REPEAT
      if predictedSplit(a:b) == acutalSplit(a:b)
          messageBits(i)=1;
      else
          messageBits(i)=0;
      end
      i+=1; a+=5; b+=5;
    UNTIL 4 message bits are extracted;
    embeddedMessage =
contatinate(embeddedMessage, messageBits);
  end // if !skipFlag
UNTIL all CUs are decoded

RETURN embeddedMessage
```

Figure 5. Pseudocode for the message embedding algorithm.

The prediction model can be regenerated by the decoder as both the feature variables and split decisions are available for both the encoder and decoder. Referring back to the example of Figure 3-'b' above, the decoder predicts the following four 32x32 CU split patterns: [01011][00001][00001][00011]. The decoder will also decode the following split flags from the bit stream (Figure 3-'c'): [00011][01001][00001][00011]. Since the last two 32x32 CUs have the same predicted and bit stream split flags then the embedded message is 0011.

In the next section, we explain the details of feature extraction and model generation proposed in this work.

3. Model generation

The model generation requires feature variables and corresponding split flags as illustrated in Figure 1. A mapping between the two is then computed. As mentioned in the previous section, the features are collected from five surrounding CUs that are previously encoded. The feature variables should be available for both the encoder and the decoder as both will compute the model weights. The feature variables are listed in Table 1.

Table 1 List of feature variables used in model generation

Feature variables	Count
Mean and variance [MVx, MVy] of list0 of surrounding 5 CUs	20
Mean and variance [MVx, MVy] of list1 of surrounding 5 CUs	20
Mean and variance of depth of surrounding 5 CUs	10

For each 64x64 CU, the encoder stores the feature variables in a feature vector and also stores the corresponding split decisions. For the model generation, the objective is to find a mapping between the feature vectors and the split decisions. In this paper we formulate this problem using Minimum Mean Square Error (MMSE).

Denote by $\mathbf{X}=[\mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_n]$ the sequence of feature vectors where $\mathbf{X}\in\Re^{m\times n}$, m is the dimensionality of the feature vector, and n is the total number of feature vectors. The corresponding CU split decisions are denoted by $\mathbf{S}=[\mathbf{s}_1, \mathbf{s}_2, \cdots, \mathbf{s}_n]$ where $\mathbf{S}\in\Re^{l\times n}$, l is the dimensionality of the split patterns and n is the total number of CUs. Model weights are computed by minimizing the weighted mean square error between the predicted and desired split decisions, $\arg\min_{\mathbf{H}}(\mathbf{HX}-\mathbf{S})(\mathbf{HX}-\mathbf{S})^T$.

The closed form solution is given by [19]:

$$\mathbf{H}=\mathbf{S}\mathbf{X}^T(\mathbf{X}\mathbf{X}^T)^{-1} \quad (1)$$

Where $\mathbf{H}\in\Re^{l\times m}$ contains the model weights.

To predict the split decisions of a feature vector $\mathbf{x}_i$, the weights obtained from (1) are used in (2)

Shanableh, T. Multimed Tools Appl (2017). doi:10.1007/s11042-017-4787-6
The final publication is available at Springer via http://dx.doi.org/10.1007/s11042-017-4787-6

$$\widehat{\mathbf{s}_i} = \mathbf{H}\mathbf{x}_i \qquad (2)$$

Where $\widehat{\mathbf{s}_i}$ is a vector of length l which means that the whole split decisions are predicted at once. he predicted split decisions are clipped to the range of 0,1 and rounded to either 0 or 1.

A cleanup process follows to make sure that there is no contradiction between the 16x16, 32x32 and 64x64 split flags. Namely, if any of the sub CU split flags is "1" then the higher level CU split flag will be set to "1" as well.

Such MMSE prediction approach was successfully used in image processing applications such as the prediction of high frequency DCT coefficients in super image resolution as reported in [19].

The predicted split decisions can then be used in the process of message embedding as illustrated in Figure 2. They are also used in the process of message extraction as illustrated in Figure 3.

It is worth mentioning that predicting split flags or decisions is also reported in the literature for the purpose of controlling the complexity of the encoding process [20], [21] and [22]. Predicting the split of intra-only CUs is also reported in [23] and [24].

The difference between these solutions and the proposed technique of split prediction is due to two factors. First, the prediction of the split decisions in this work is based on feature variables extracted from previously encoded CUs. This is important as the prediction model needs to be regenerated at the decoder. Without regenerating the predication model, the decoder will not be able to identify the sub CUs that contain the embedded message. Second, the prediction solution is capable of predicting all split decisions without the need to extracted further features at 64x64, 32x32 and 16x16 sub CU levels.

On the other hand in existing work, feature variables are based on information available to the encoder but not to the decoder. Therefore, the prediction model cannot be regenerated at the decoder. Furthermore, the feature variables are selected at 3 CU levels; 64x64, 32x32 and 16x16. Therefore, the prediction of split decisions cannot be performed at the highest CU level. This also implies that the decoder cannot regenerate the model.

4. Experimental results

In the experiments to follow and for a fair comparison with [16], we use the same video sequences and same experimental setup. As mentioned in the introduction, the reviewed work embeds a message in HEVC video by forcing certain partitioning types for the Prediction Units (PUs). If a message bit is '0' then the PU type is restricted to: NxN, nLx2N, 2NxnU and 2NxN. And if the message bit is '1'then the PU type is restricted to: 2Nx2N, nRx2N, Nx2N and 2NxnD.

The video sequences used are Tennis 1920×1080@24Hz, FourPeople 280×720@60Hz, BasketballDrill 832×480@50Hz and asketballPass 416×240@50Hz. The video sequences are coded with the following constant bit rate values 100Kb/s, 500Kb/s, 1Mb/s, 5Mb/s and 10Mb/s. The message to be embedded is generated using a uniform discrete binary number generator.

We use the HEVC reference software HM13.0 [25]. The main profile with a level of 3.1 is used. The video coding structure is IPPP… using 4 reference frames. The maximum CU size is 64x64 pixels. The asymmetric motion partitions tool and the adaptive loop filter tool are both enabled. The first 10% of frames of each sequence are used for model generation.

We start by examining the prediction accuracy of the split decisions as proposed in Section 3. The total number of correctly predicted split decisions divided by the total number of split decisions is reported in Table 2. The results are the average prediction accuracy for the entire above-mentioned bit rates.

Table 2 Prediction accuracy of split decisions.

Sequence	Prediction accuracy %
BasketballPass	82.2
BasketballDrill	86.7
FourPeople	90.1
Tennis	93.4
Average	88.1

It is shown in the table that the prediction accuracy ranges from 82% to 93%. The average accuracy is 88.1%. This indicates that the proposed prediction solution is capable of predicting split decisions with a very good accuracy. The next step is to use this prediction approach in message embedding and analyze the results in terms of the payload of the embedded message in Kbit/s and the drop in PSNR as a result of embedding. Clearly, a higher message

Shanableh, T. Multimed Tools Appl (2017). doi:10.1007/s11042-017-4787-6
The final publication is available at Springer via http://dx.doi.org/10.1007/s11042-017-4787-6

payload is preferred and lower drop in PSNR is also preferred. The drop in PSNR is computed as the difference between the PSNR of video coding with and without message embedding.

In the first set of results reported in Table 3, we compare between the results of message embedding into 16x16 and 32x32 split decisions.

Table 3 Total message payload and drop in PSNR, comparison between two proposed solutions.

	Mb/s	Modify 16x16 Splits		Modify 32x32 Splits	
		PSNR loss dB	Msg Kb/s	PSNR loss dB	Msg Kb/s
Tennis 1920×1080 @24Hz	0.1	1.15	58.5	0.49	14.6
	0.5	5.1	109.7	1.45	34.6
	1	2.9	150	0.76	37.3
	5	0.6	192.5	0.23	48.1
	10	0.27	200.1	0.13	48.9
Average		2.0	142.2	0.6	36.7
FourPeople 1280×720 @60Hz	0.1	1.5	43.4	1.3	10.9
	0.5	1.3	76.5	0.5	19.1
	1	0.63	89.2	0.33	22.3
	5	0.13	140.4	0.1	35.1
	10	0.1	170.9	0.09	43
Average		0.7	104	0.46	26.1
Basketball-Drill 832×480 @50Hz	0.1	2.2	32.2	1.67	8.1
	0.5	1.42	54	0.6	13.5
	1	0.9	60.1	0.48	15.4
	5	0.53	73	0.36	18.3
	10	0.29	74.3	0.2	18.6
Average		1.1	58.7	0.66	14.8
Basketball-Pass 416×240 @50Hz	0.1	1.9	15.6	0.7	3.9
	0.5	0.8	17.8	0.54	4.5
	1	0.62	18.4	0.47	4.6
Average		1.1	17.3	0.57	4.3
Overall average		1.24	87.6	0.58	22.3

Examining the overall averages in the table, it is shown that the average message payload as a result of modifying 16x16 split decisions is 87.6Kbit/s whereas that of modifying 32x32 split decisions is 22.3 Kbit/s. The former solution embeds a message payload 4 times as much the latter solution. This is expected as each 32x32 CU contains four 16x16 sub CUs. Clearly this comes at a disadvantage of further reduction in the overall PSNR. In the table it is shown that the average drop in PSNR due to modifying 16x16 split decisions is1.24dB whereas that of modifying the 32x32 split decisions is 2.23dB.

The adverse effect of the message payload on the video quality is summarized in Figure 7.

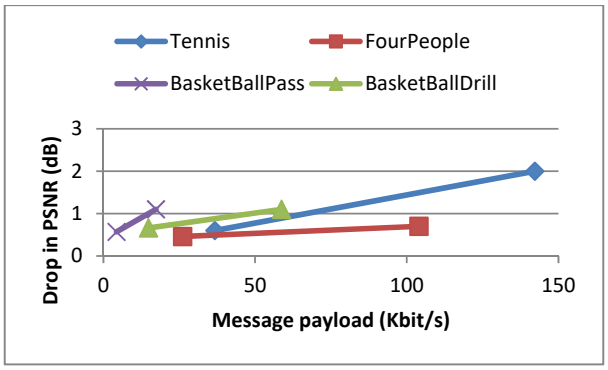

Figure 7. Effect of message payload on video quality

Note that the message payload depends on the CU split level at which data is embedded. In this work, two solutions are proposed; message embedding at 32x32 and message embedding at 16x16 CU depths. Therefore, for each test sequence, two data points are available as shown in Figure 7.

Further investigation into the statistics of message embedding is reported in Table 4. We report the total number of modified split decisions in both of the proposed solutions. This is reported as the number of modified split decisions per second of video in Kbit/s, where each split decisions is represented as a bit.

Table 4 Total number of modified split decisions in two proposed solutions.

Shanableh, T. Multimed Tools Appl (2017). doi:10.1007/s11042-017-4787-6
The final publication is available at Springer via http://dx.doi.org/10.1007/s11042-017-4787-6

Count of modified splits (Kbit/s)			
	Mb/s	Modify 16x16 Splits	Modify 32x32 Splits
Tennis 1920×1080 @24Hz	0.1	29.2	14.6
	0.5	54.8	27.4
	1	75	37.5
	5	96.1	48.1
	10	100.3	50.7
Average		71.1	35.7
FourPeople 1280×720 @60Hz	0.1	21.7	10.9
	0.5	38.2	19.1
	1	44.6	22.3
	5	70.2	35.1
	10	85.5	42.7
Average		52.0	26.0
Basketball-Drill 832×480 @50Hz	0.1	16.1	8.1
	0.5	27	13.5
	1	30.2	15.1
	5	36.5	18.3
	10	37.1	18.6
Average		29.4	14.7
Basketball-Pass 416×240 @50Hz	0.1	7.8	3.9
	0.5	8.9	4.5
	1	9.2	4.6
Average		8.6	4.3
Overall average		43.8	21.9

For the proposed solution of modifying 16x16 split decisions, it is shown that the amount of modified split decisions is half of the corresponding message payload reported in Table 3. This is so because message bits are embedded into 16x16 split flags and the message is generated with a uniform distribution of '1s' and '0s' which is the worst case scenario in terms in message embedding. Therefore it is expected that, in the worst case scenario, half of the 16x16 split decisions are modified.

As for the second solution of message embedding into 32x32 split decisions, the conclusion is different. In this solution, a 32x32 CU contain four 16x16 split flags. Message embedding is done by leaving the 32x32 split decisions as is or by modifying 1 or more of its 16x16 split decisions. Hence in the worst case scenario, four split decisions might be modified for embedding one message bit. Therefore, when comparing the number of modified split decisions in this solution to the average message payload reported in Table 3, it is not necessarily half the value. In fact the average message payload is 22.3Kbit/s and the average modified split decisions is 21.9Kbit/s.

In summary, comparing the averages of Tables 3 and 4 it is revealed that embedding 2 message bits by modifying the 16x16 split decisions require modifying one split decisions on average. Whereas embedding 2 message bits by modifying the 32x32 split decisions require modifying 2 split decisions on average.

Lastly, in Table 6, we compare the proposed solution of message embedding into 16x16 split flags with existing work as reported in [16]. As mentioned in the introduction, the reviewed solution embeds a message bit in every Partition Unit (PU) of the 64x64 CU. The embedding is based on modifying the corresponding PU types.

Table 5 Total message payload and drop in PSNR, comparison with existing solution.

	Mb/s	Proposed(16x16)		Reviewed [16]	
		PSNR loss dB	Msg Kb/s	PSNR loss dB	Msg Kb/s
Tennis 1920×1080 @24Hz	0.1	1.15	58.5	5.34	144.4
	0.5	5.1	109.7	8.21	146.3
	1	2.9	150	5.44	147.2
	5	0.6	192.5	0.95	188.2
	10	0.27	200.1	0.51	223.2
Average		2.0	142.2	4.1	170
FourPeople 1280×720 @60Hz	0.1	1.5	43.4	4.66	61.8
	0.5	1.3	76.5	4.78	61.9
	1	0.63	89.2	1.26	66.6
	5	0.13	140.4	0.22	94.8
	10	0.1	170.9	0.18	116.1
Average		0.7	104	2.2	80.2
Basketball-Drill 832×480 @50Hz	0.1	2.2	32.2	2.54	26
	0.5	1.42	54	1.65	30.1
	1	0.9	60.1	1.09	38.5
	5	0.53	73	0.71	59.2
	10	0.29	74.3	0.55	69.9
Average		1.1	58.7	1.3	44.7
Basketball-Pass 416×240 @50Hz	0.1	1.9	15.6	2.25	7.8
	0.5	0.8	17.8	0.95	13.9
	1	0.62	18.4	0.76	17.1
Average		1.1	17.3	1.3	12.9
Overall average		1.24	87.6	2.23	77

It is shown in the table that the distortion caused by the proposed solution is consistently lower than that of the reviewed solution. It is also shown that the message payload of the proposed system is on average 87.6 Kbit/s whereas that of the reviewed system is 77Kbit/s. The reviewed work has a high

message payload as well, however, imposing restrictions on PU types seem to cause an adverse effect on video quality.

These results are a good indication that message embedding by altering the split decisions of CU using the proposed prediction approach results in a high message payload with a reduced effect on video quality.

5. Conclusion

A solution was proposed for embedding messages into HEVC coded video at a syntax-level. The split decisions of 32x32 and 16x16 CUs are modified according to three inputs; the message bits, true split flags and predicted split flags. The encoder starts by generating a set of model weights for predicting the split flags of all split decisions of a 64x64 CU. The decoder repeats the same process and re-generates the weights. Message extraction is then performed by comparing the predicted split decisions to the true split decisions received in the video bit stream. Experimental results revealed that embedding 2 message bits by modifying the 16x16 split decisions require modifying one split decisions on average. Whereas embedding 2 message bits by modifying the 32x32 split decisions require modifying 2 split decisions on average. Experiments performed on various video sequences with different resolutions resulted in an average message payload of 87.6 Kbit/s, achieved with a PSNR degradation of 1.24 dB. Whereas, the two measures for the reviewed work are 77 Kbit/s and 2.23 dB respectively.

References

[1] L. Tian, N. Zheng, J. Xue, C. Li, X. Wang, "An integrated visual saliency-based watermarking approach for synchronous image authentication and copyright protection," Signal Processing: Image Communication, 26(8-9), pp. 427-437, October, 2011.

[2] F.C. Chang, H.C. Huang and H.M. Hang, "Layered access control schemes on watermarked scalable media," Journal of VLSI Signal Processing, 49(2007), pp. 443–455, 2007.

[3] P.C. Su, C.-S. Wu, I.-F. Chen, C.-Y. Wu, Y.-C. Wu, "A practical design of digital video watermarking in H.264/AVC for content authentication," Signal Processing: Image Communication, 26(8-9), pp. 413-426, October, 2011.

[4] S. Emmanuel, A. Vinod, D. Rajan, C.K Heng, "An Authentication Watermarking Scheme with Transaction Tracking Enabled," In Proc. Digital EcoSystems and Technologies Conference, 2007. Inaugural, 21-23 February, 2007.

[5] S. Kapotas, A. Skodras, "A new data hiding scheme for scene change detection in H.264 encoded video sequences," IEEE International Conference on Multimedia and Expo ICME 2008, pp.277-280, June, 2008.

[6] A. Yilmaz, A. Aydin, "Error detection and concealment for video transmission using information hiding," Signal Processing: Image Communication, 23(4), pp. 298-312, April, 2008.

[7]D. Xu, R. Wang and Y. Q. Shi, "Data Hiding in Encrypted H.264/AVC Video Streams by Codeword Substitution," IEEE Transactions on Information Forensics and Security, 9(4), pp. 596-606, April, 2014.

[8]T. Shanableh, "Matrix encoding for data hiding using multilayer video coding and transcoding solutions," Signal Processing: Image Communication, Elsevier, 27(9), pp. 1025-1034, October, 2012.

[9] T. Shanableh, "Data Hiding in MPEG Video Files Using Multivariate Regression and Flexible Macroblock Ordering," IEEE Transactions on Information Forensics and Security, 7(2), pp.455-464, April, 2012.

[10] T. Stütz, F. Autrusseau and A. Uhl, "Non-Blind Structure-Preserving Substitution Watermarking of H.264/CAVLC Inter-Frames," IEEE Transactions on Multimedia, 16(5), pp. 1337-1349, Aug. 2014.

[11] Y. Cao, H. Zhang, X. Zhao and H. Yu, "Covert Communication by Compressed Videos Exploiting the Uncertainty of Motion Estimation," IEEE Communications Letters, 19(2), pp. 203-206, February, 2015.

[12] K. Wang, H. Zhao and H. Wang, "Video Steganalysis Against Motion Vector-Based Steganography by Adding or Subtracting One Motion Vector Value," IEEE Transactions on Information Forensics and Security, 9(5), 741-751, May 2014.

[13] K. Tasdemir, F. Kurugollu and S. Sezer, "Spatio-Temporal Rich Model-Based Video Steganalysis on Cross Sections of Motion Vector Planes," IEEE Transactions on Image Processing, 25(7), pp. 3316-3328, July, 2016.

[14] Y. Hu, C. Zhang and Y. Su, "Information Hiding Based on Intra Prediction Modes H.264/AVC," IEEE International Conference on Multimedia and Expo, ICME 2007, pp.1231-1234, July, 2007.

[15] G. Yang, J. Li, Y. He and Z. Kang, "An information hiding algorithm based on intra-prediction modes and matrix coding for H.264/AVC video stream," International Journal of Electronics and Communications, (65)4, pp. 331-337, April, 2011.

[16] Yiqi Tew, KokSheik Wong "Information hiding in HEVC standard using adaptive coding block size decision," IEEE International Conference on Image Processing, ICIP 2014, Paris, France, October, 2014.

[17] J. Wang, R. Wang W. Li ,D. Xu and M. Huang, "An Information Hiding Algorithm for HEVC

Based on Intra Prediction Mode and Block Code," Sensors & Transducers, 177(8), pp. 230-237, August, 2014.
[18] E. Peixoto, T. Shanableh and E. Izquierdo, "H.264/AVC to HEVC Video Transcoder based on Dynamic Thresholding and Content Modeling, " IEEE Transactions on Circuits and Systems for Video Technology, 24(1), January, 2014.
[19] K.W. Hung and W.C. Siu, "Novel DCT-Based Image Up-Sampling Using Learning-Based Adaptive K-NN MMSE Estimation," IEEE Transactions on Circuits and Systems for Video Technology, 24(12), pp.2018-2033, December, 2014.
[20] H.-S. Kim and R.-H. Park , "Fast CU Partitioning Algorithm for HEVC Using an Online-Learning-Based Bayesian Decision Rule," IEEE transactions on circuits and systems for video technology, 26(1), January, 2016.
[21] G. Correa, P. A. Assuncao, L. V. Agostini, and L. A. da Silva Cruz, "Fast HEVC Encoding Decisions Using Data Mining," IEEE transactions on circuits and systems for video technology, 25(4), April, 2015.
[22] A. Heindel, T. Haubner and A. Kaup, "Fast CU Split Decisions for HEVC Inter Coding Using Support Vector Machines," Processing of Picture Coding Symposium, Germany, December, 2016.
[23] X. Zheng, Y. Zhao, H. Bai and C. Lin, "Fast Algorithm for Intra Prediction of HEVC Using Adaptive Decision Trees," KSII Transactions on Internet and Information Systems, 10(7), pp. 3286-3300, 2016.
[24] S. Cho and M. Kim, "Fast CU Splitting and Pruning for Suboptimal CU Partitioning in HEVC Intra Coding," IEEE Transactions on Circuits and Systems for Video Technology, 23(9), September, 2013.
[25] I.-K. Kim, K. D. McCann, K. Sugimoto, B. Bross, W.-J. Han and G. J. Sullivan, "High Efficiency Video Coding (HEVC) Test Model 13 (HM13) Encoder Description," Document: JCTVC-O1002, Joint Collaborative Team on Video Coding (JCT-VC) of ITU-T SG16 WP3 and ISO/IEC JTC1/SC29/WG11, 15th Meeting: Geneva, CH, 23 Oct. – 1 November, 2013

Tamer Shanableh received his Ph.D. in Electronic Systems Engineering in 2002 from the University of Essex, UK. From 1998 to 2001, he was a senior research officer at the University of Essex, during which, he collaborated with BTexact on inventing video transcoders. He joined Motorola UK Research Labs in 2001. During his affiliation with Motorola, he contributed to establishing a new profile within the ISO/IEC MPEG-4 known as the Error Resilient Simple Scalable Profile. He joined the American University of Sharjah in 2002 and is currently a professor of computer science. Dr. Shanableh spent the summers of 2003, 2004, 2006, 2007 and 2008 as a visiting professor at Motorola multimedia Labs. He spent the spring semester of 2012 as a visiting academic at the Multimedia and Computer Vision and Lab at the School of Electronic Engineering and Computer Science, Queen Mary, University of London, London, U.K . His research interests include digital video processing and pattern recognition.