IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received August 2, 2016, accepted September 28, 2016, date of publication October 21, 2016, date of current version November 18, 2016.

Digital Object Identifier 10.1109/ACCESS.2016.2619259

Extrinsic Information Modification in the Turbo Decoder by Exploiting Source Redundancies for HEVC Video Transmitted Over a Mobile Channel

RYAN PERERA[1], HEMANTHA KODIKARA ARACHCHI[1], (Member, IEEE), MUHAMMAD ALI IMRAN[2], (Senior Member, IEEE), AND PEI XIAO[3], (Senior Member, IEEE)

[1]CVSSP, University of Surrey, Guildford, GU2 7XH, U.K.
[2]School of Engineering, University of Glasgow, Glasgow, G12 8QQ, U.K.
[3]ICS, University of Surrey, Guildford, GU2 7XH, U.K.

Corresponding author: R. Perera (g.perera@surrey.ac.uk)

This work was supported by the U.K. Engineering and Physical Sciences Research Council under Grant EP/N020391/1.

ABSTRACT An iterative turbo decoder-based cross layer error recovery scheme for compressed video is presented in this paper. The soft information exchanged between two convolutional decoders is reinforced both by channel coded parity and video compression syntactical information. An algorithm to identify the video frame boundaries in corrupted compressed sequences is formulated. This paper continues to propose algorithms to deduce the correct values for selected fields in the compressed stream. Modifying the turbo extrinsic information using these corrections acts as reinforcements in the turbo decoding iterative process. The optimal number of turbo iterations suitable for the proposed system model is derived using EXIT charts. Simulation results reveal that a transmission power saving of 2.28% can be achieved using the proposed methodology. Contrary to typical joint cross layer decoding schemes, the additional resource requirement is minimal, since the proposed decoding cycle does not involve the decompression function.

INDEX TERMS Combined source channel coding, EXIT charts, iterative decoding, mobile communication, turbo codes, video compression.

I. INTRODUCTION

In a survey conducted by Qualcomm Technologies in 2013 [1], three of the top seven features new buyers seek in a mobile device are related to the viewing experience such as display size, display quality and display resolution. This trend is also evidenced by the tremendous burst of demand for video data we have seen over the last few years over mobile networks, that mobile video accounted for 55% of mobile data traffic during 2014 [2]. Scientists seeking to accommodate this demand strive to compress video data to the greatest extent possible [3], while networking engineers strive to deliver these data as reliably as possible. As such, we have seen novel radio resource allocation methods [4], [5], intelligent adaptive modulation and coding techniques [6] and numerous forward error correcting methods, such as turbo codes [7], performing very close to the theoretical channel capacity.

Among the efforts of finding efficient means for compressed mobile video delivery, improved video recovery by means of joint source-channel decoding (JSCD) is gaining interest among researchers. JSCD is a cross layer approach, in which data flows between the application layer (APP) video compression source decoder and the physical layer (PHY) channel decoder to collaboratively recover errors. For any form of error recovery, the available redundancy in the received bit stream must be leveraged, be it in the form of controlled channel code redundancy or the residual source redundancy that survived the compression stage. The receiver is more equipped for error recovery when more redundancy is available.

Reference [8] gives an insight into ways of quantifying source code redundancy and channel code redundancy of transmission-ready H.263-compressed video data. It pointed out the fact that not all binary patterns are legitimate code word sequences, and not all code word sequences refer to a legitimate picture block, inferring the existence of a significant amount of residual source redundancy. For instance, the number of redundancy bits in a H.263 compressed image

block, whose length is typically around 60 bits, is 11 bits [8]. However, the source redundancy exploitation methodology presented in [8] is computationally very expensive and obsolete for today's video transmission schemes, where more efficient entropy coding schemes such as Context Adaptive Binary Arithmetic Coding (CABAC) [9] are deployed in place of run-length variable length code (VLC).

Exploiting the source semantics of variable length codes in H.263 [10] compressed video, such as the number of DCT coefficients and the number of bits in the sequence, authors in [11] proposed a JSCD algorithm, which required only the knowledge of the length in bits of the VLC sequence as an a priori information. Abiding to the fundamentals of Viterbi algorithm [12] it selected a survivor sequence while reading the VLC sequence. The survivor selection was based on the conventional Viterbi metric, the VLC structure, and source semantics constraints on the sequence. This algorithm demonstrated an unprecedented gain of up to 30dB of PSNR after three turbo decoding iterations in comparison to a conventional H.263 decoder, which did not utilize the VLC syntax (the run-length-last triplet syntax) or source semantics; and a 16dB gain in comparison to a decoder, which used only VLC syntax constraints. However, with the introduction of HEVC we have seen the exploitation of these run-length-last VLC redundancies by the video compressor. Consequently, they are no longer at the receiver's disposal for error recovery.

The works presented in [13]–[18] attempted to correct the channel decoded and corrupted packets by flipping one or more bits to nominate several candidate patterns and passing it on to a syntax checker until a valid sequence was identified. W. E. Lynch, in [17], used the syntactic/semantic rules of compressed video to detect errors. The algorithm therein, chose a predefined number (n_F) of the smallest absolute post turbo decoder log likelihood ratio (LLR) values (which the authors referred to as flip bits) and generated a set of video packet candidates by allocating all possible bit combinations for the flip bits. The video decompression routine verified each candidate for its conformance to the syntax, modified the LLR values and fed them back to the turbo decoder. Although this was feasible for small n_F, it becomes increasingly complex (in the order of 2^{n_F}) as the channel degrades and n_F increases, a view also shared in [19]. While [17] was demonstrated for MPEG-4 compressed video, a similar approach was demonstrated for H.264 in [18]. In both references, the header stream and video packet lengths were assumed to be delivered error free, which is a highly unlikely scenario.

Another approach to reinforcing extrinsic information at the turbo decoder was presented by Z. Peng et al. in [20]. Here, reinforcements came after an APP image processing stage, and was in the form of scaling or descaling certain soft values of the last extrinsic information sequence that was fed into a constituent MAP decoder. The authors' focus was mainly on recovering vector quantization coded images [21], where their most significant gains were demonstrated. When compressed using vector quantization, a block error in the receiver reconstructed image can be directly attributed to a specific segment in the bit stream that was the output from the turbo decoder. After detecting the erroneous blocks using a boundary matching based block error detection algorithm, they modified the a priori probabilities of the bit positions (the extrinsic information) that was fed back into the turbo decoder. The latest compression standard HEVC however, does not portray such an identifiable mapping between the image blocks and the compressed bit stream as in the case of vector quantization coded images.

In the case of HEVC, recovery of the slice header is much more important than the recovery of slice data. Most literature, [11], [17], [18], [20], deemed the use of very strong coding schemes to protect the header stream such that error-free transmission can be assumed. Employing such a strong coding scheme would overshadow any gain achieved by the proposed methods. We propose a technique to recover the header information using the syntactical conformance verification and soft information turbo decoding. The methodology is demonstrated for the state of the art video compression scheme HEVC. Since HEVC has only recently been introduced commercially, literature attempting a JSCD approach for HEVC compressed video recovery is very scarce. However, for comparison purposes we adopt the Peng's implementation on MPEG coded video [20], which like HEVC, portrays a well-organized bit structure.

This paper is organized as follows. Section II introduces the system model and assumptions. Focus is narrowed in Section III on the receiver operations, along with notations defined. Section IV presents a mandatory prerequisite identification for the success of the proposed algorithm. Details on header field correction are included in Section V. Section VI demonstrates the performance of the technology in realistic contexts. Section VII concludes the paper.

II. JOINT SOURCE CHANNEL DECODING

In the receiver's attempt to recover the compressed video data, the receiver must make use of the redundancy available within the bit stream. This redundancy can be the residual, unintentional redundancy that survive the video compression stage or the intentionally added redundancy at the channel coding stage. The traditional approach to video recovery is error correction using channel redundancy, followed by video decompression, and then error concealment using source redundancy. Nevertheless, research [22] has found that these two types of redundancy can be used collaboratively to improve the overall performance in video recovery. This section introduces a system model that can be used for such a collaborative approach, and introduces the assumptions applicable for the proposed algorithm.

In order to improve the turbo code performance, one can either alter its constituent convolutional decoders, or reinforce the extrinsic information exchanged between the decoders. The approach proposed in this paper considers the latter approach. A mobile receiver which attempts to recover compressed video data after being received over a noisy wireless channel is considered. An overview of the system

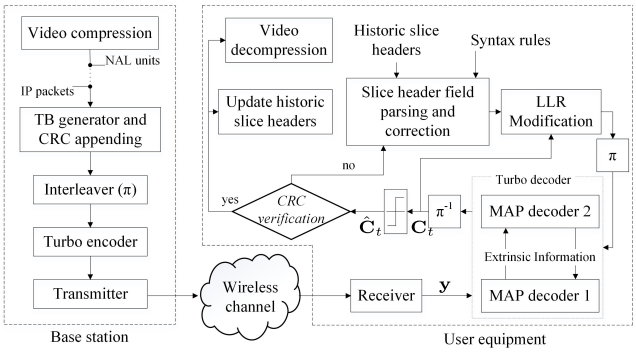


FIGURE 1. End-to-end overview of the proposed system.

model is depicted in Fig. 1. For reinforcing the extrinsic information of the turbo decoder, the Slice-header-field Parsing and Correction (SPC) module plays a pivotal role. The *historic slice headers*, which store past decoded information of the video, and the *syntax rules*, which impart knowledge on how an HEVC compliant bit stream is structured, assist in the correction function.

The output from the turbo decoder can be any pattern of bits if the transmission experiences noisy channel conditions. Many of these patterns result in invalid HEVC semantics and are illegitimate compressed bit sequences. Similar behavior has been observed in [8] for previous video coding standards such as H.263. The SPC module checks the HEVC syntax of the headers, and if an error is identified, it attempts to deduce the correct header. Based on this alteration the LLR stream is modified. These soft values are then fed back to the turbo decoder as extrinsic information. This is the basic premise behind the innovation presented in this paper. As an overview, the proposed JSCD approach turbo decodes the data initially, modifies the a posteriori LLR values based on the source semantics and other factors, and recommences turbo decoding. This is an iterative operation between turbo iterations and LLR modification until the errors are corrected.

A. HEVC STANDARD

Most video compression standards [3], [10], [23] divide each frame into blocks and apply a hybrid approach of inter-/intra prediction and 2-D transform coding. In the case of HEVC, the prediction and 2-D transformation is performed on slices, an area in the picture that can be decoded independent of other slices of the same picture. The compressed slices are encapsulated into Network Abstraction Layer (NAL) units, which contain headers and a section of CABAC coded slice data. The CABAC coded section is characterized by a plethora of possible bit patterns, and detecting, let alone correcting a faulty bit is hardly viable. On the contrary, the more important slice headers follow a rigid set of rules and exhibit a higher degree of source redundancy in the form of;

- patterns recognizable between the neighboring headers,
- the clearly defined field structure within the headers, and
- the repetitions among header fields within a picture frame.

Due to their organized structure and the severe consequences of their deformation on the decompression function, our focus is only on rectifying the bit positions of the slice segment headers and NAL unit headers. The slice header syntax follows the criteria given in [24, Sec. 7.3.6.1]. Exploitation of the aforementioned forms of redundancies benefit only the header regions, and therefore, a mechanism is required to spread the benefits across the remaining portion of the bit sequence.

B. HEVC OVER LTE-ADVANCED

In this paper, it is considered that each NAL unit occupies one IP packet. This is the preferred and most efficient choice for packetized transmission [25]. Since the IP packet size is restricted, the NAL unit size is restricted, and the area of the slice often varies to fully occupy the permitted NAL unit size. The overhead burden, as the APP payload reaches the PHY transport block (TB) generator, is in the form of headers only (e.g., IP and MAC addresses, timestamp, checksums, payload type). The input to the PHY is in the form of TBs entailing one or more IP packets. The PHY functions closely follow the 3GPP standard for Long Term Evolution (LTE) advanced [26], [27].

With the inclusion of an interleaver before the turbo encoder, the header bits are placed among the CABAC coded bits. Therefore, at the turbo decoding stage after the LLR modification process, the CABAC coded bits will also benefit from the alterations made. Note that this inclusion is in addition to the interleaving function between the turbo encoder and the transmitter, already specified in the LTE-Advanced standard. In the event of error bursts, the standard-compliant interleaver ensures that errors are distributed between slices, so the turbo decoder and the source decoder are better equipped to recover the NAL units. However, it does not solve the problem that the LLR modification operation only improves the header regions. Therefore, an interleaver preceding the turbo encoder is necessary for the proper functioning of the proposed JSCD approach.

Sections III-V present the main contributions of this paper. An algorithm for access unit boundary identification in a corrupted bit sequence is presented. Subsequently, algorithms are introduced to analyze and amend NAL unit header fields, thus altering the extrinsic information exchanged within the turbo decoder. These algorithms are utilized in the iterative model indicated in Fig. 1.

III. RECEIVER OPERATIONS

After performing the reverse PHY operations at the receiver, the bit stream is passed on to the turbo decoder. Initially, the sequence of a priori LLR values, which are used in the BCJR[1] algorithm, is set to an all-zero vector. The turbo decoder iterates until either the transport block cyclic redundancy check (CRC) is successful or until the maximum number

[1]Named after the four founders of the algorithm; Bahl, Cocke, Jelinek and Raviv.

of turbo iterations is reached. The choice for the maximum number of turbo iterations is based on EXIT charts [28], which will be further discussed in Section VI. Faulty transport blocks are compared against the video compression syntax and modifications are performed within the SPC module and LLR modification module. Subsequently, this modified LLR stream is input back to the turbo decoder, as the a priori LLR sequence, to perform the recursive BCJR algorithm. Transport blocks characterizing a successful CRC are forwarded to the video decompression stage and to the *historic slice headers*.

Denoting the k^{th} transmitted bit as u_k, the soft decision on u_k at the output of an MAP decoder is given by

$$\Lambda(u_k) = L(u_k) + y_k^s + L_e(u_k) \tag{1}$$

where

$L(u_k)$	a priori LLR of u_k,
y_k^s	received value for the k^{th} systematic bit,
$L_e(u_k)$	extrinsic information conveyed to the other decoder.

All terms in (1) are log likelihood ratios. $L_e\,(u_k)$ is derived using the BCJR algorithm, and for the subsequent MAP decoder, $L\,(u_k) = L_e\,(u_k)$. As the turbo iterations progress, the magnitude of $L_e\,(u_k)$ values increase, indicating the rise in reliability. By altering $L\,(u_k)$ based on external source syntax information, $L_e\,(u_k)$ of the subsequent MAP decoders are updated and the decoding process is effectively reinforced (note that for the first MAP decoder, immediately after the LLR modification module $L_e\,(u_k)$ is reset to an all-zero vector).

A. DEFINITIONS AND NOTATIONS

The a posteriori log likelihood ratio after the turbo decoder is denoted as $L(u[k])$. This relates to the k^{th} bit in the transport block of size T bits. After the hard decision, a sequence of T bits is forwarded to the SPC module. The hard decision of the k^{th} bit is denoted as $\hat{u}[k]$. Let's consider the t^{th} transport block since the beginning of the considered video sequence transmission. The LLR values of this t^{th} transport block are:

$$\mathbf{C}_t = \{L\,(u[k]) \mid k \in [(t-1)T+1, tT]\} \tag{2}$$

The hard decision values on the t^{th} transport block are:

$$\hat{\mathbf{C}}_t = \left\{\hat{u}\,[k] \mid k \in [(t-1)T+1, tT]\right\} \tag{3}$$

The task is to modify (2) in a manner to comply with the HEVC syntax and follows the trends set by the preceding bits $\left\{\hat{u}[k] \mid k \in [1, (t-1)T]\right\}$.

Next, the notation for the complete transmitted video sequence is defined. Denoting $\mathbf{N}_i$ as the bit sequence of the i^{th} IP packet, the sequence of cumulative IP packet lengths can be denoted by $\mathbf{L} = \left\{l_i = \sum_{m=1}^{i} \text{length}(\mathbf{N}_m) \mid i \in [1, N]\right\}$, where N is the total number of IP packets in the transmission, and length($\mathbf{N}_m$) indicates the number of bits in $\mathbf{N}_m$. The set of LLR values pertaining to the complete stream of IP packet bits is $\{L\,(u[k]) \mid k \in [1, l_N]\}$.

The LLR values of the i^{th} IP packet are:

$$\{L\,(u[k]) \mid k \in [l_{i-1}+1, l_i]\} \quad \text{with } l_0 = 0 \tag{4}$$

For simplicity, we define $j_i = l_i + h,\ \forall i \in [1, N]$, where h is the number of bits in the IP packet header overhead. Then, $(j_{i-1}+1)$ is the starting index of the i^{th} NAL unit.

Denote the first starting bit of a NAL unit in $\hat{\mathbf{C}}_t$ as $\hat{u}[j_{i-1}+1]$. Note that the initial part of $\hat{\mathbf{C}}_t$ may contain a partial NAL unit, and $\hat{u}[j_{i-1}+1]$ is the starting bit of the NAL unit that follows the partial NAL unit. $\hat{u}[j_{i-1}+1]$ belongs to the i^{th} NAL unit in the video transmission. In the rest of the paper, it is also assumed that the last portion of $\hat{\mathbf{C}}_t$ belongs to the $(i+n)^{\text{th}}$ NAL unit.

IV. ACCESS UNIT BOUNDARIES

Before correcting the header fields of a NAL unit, it is necessary to cluster the NAL units into their respective access units. There is a dedicated flag in the slice segment header indicating the access unit boundary: *first_slice_segment_in_pic_flag*. However, in the event of erroneous transmissions, the accuracy of this flag is not guaranteed. This section proposes a more reliable mechanism to identify the first NAL unit in each access unit. This identification is performed in the SPC module depicted in Fig. 1.

Initially, a heuristic approach is adopted to rectify the sparse bit errors using neighboring bits. The fact that certain fields (of NAL units within an access unit) remain unaltered is used in this regard. Note that the modifications performed in this section is an interim stage, the results of which are discarded after access unit boundaries are identified.

Denote $\mathbf{B}$ and $\hat{\mathbf{B}}$, bit sequences, whose i^{th} elements are b_i and $\hat{b}_i$ respectively. A function $f_a : \mathbf{B} \rightarrow \hat{\mathbf{B}}$ is defined for $\mathbf{B}$, whose elements are from the alphabet $\{-1, +1\}$, as:

$$\hat{b}_i = f_a(b_i) = \begin{cases} b_i, & \text{if } m = 0 \\ \text{sgn}(m), & \text{otherwise} \end{cases} \tag{5}$$

Here $m = 2b_i + b_{i-1} + b_{i-2} + b_{i+1} + b_{i+2}$ and sgn$(\cdot)$ is the signum function.

The value of m is calculated such that if all the 4 neighboring bits suggest that the bit value should be altered, the change is executed. Else, if at least one of the 4 neighbors suggests the bit under consideration is correct, it is left intact. The function f_a is used to heuristically rectify the sparse bit errors of the fields indicated in Table 1. These fields are chosen because they have a definite position within the NAL unit. Refer to [24] for the order and methodology of interpreting the headers.

Denote $\mathbf{B}_k = \left\{\hat{u}[j_{i-1}+k], \hat{u}[j_i+k], ..., \hat{u}[j_{i+n-1}+k]\right\}$, where $k \in \{[2, 7] \cup [14, 16] \cup 18\}$. These collocated bit sequences are modified by performing the function f_a on $\mathbf{B}_k$. The modified sequences are denoted as $\hat{\mathbf{B}}_k$.

Next the set of candidates, $\mathbf{A}$, for the first NAL unit of each access unit are identified. This algorithm is

TABLE 1. Fields on which f_a is applied.

Field	Corresponding bits of the i^{th} NAL unit
nal_unit_type	$\{\hat{u}[j_{i-1}+2], ..., \hat{u}[j_{i-1}+7]\}$
nuh_temporal_id_plus1	$\{\hat{u}[j_{i-1}+14], ..., \hat{u}[j_{i-1}+16]\}$
First bit of slice_pic_parameter_set_id	$\hat{u}[j_{i-1}+18]$

Algorithm 1 Obtaining Candidates for Access Unit Boundaries

Require: the availability of sequences $\hat{\mathbf{B}}_k$ where $k \in \{[2,7] \cup [14,16] \cup 18\}$
Ensure: all actual first NAL units are included in $\mathbf{A}$
  for $m \in [i, i+n]$ **do** ▷ NAL units in $\hat{\mathbf{C}}_t$
    $S_m \leftarrow 0$
    for $k \in \{[2,7] \cup [14,16] \cup 18\}$ **do**
      if $\hat{u}[j_{m-2}+k] \neq \hat{u}[j_{m-1}+k]$ **then**
        $S_m \leftarrow S_m + 1$
      end if
    end for
    if $\hat{u}[j_{m-1}+17] = 1$ **then**
      $S_m \leftarrow S_m + 1$
    end if
  end for
  $\mathbf{A} \leftarrow \{m | S_m \geq 3\}$

explained below and summarized in Algorithm 1. In $\hat{\mathbf{B}}_k = \{\hat{u}[j_{i-1}+k], \hat{u}[j_i+k], ..., \hat{u}[j_{i+n-1}+k]\}$, if,

$$\hat{u}[j_{m-2}+k] \neq \hat{u}[j_{m-1}+k], \text{ where } m \in [i, i+n] \quad (6)$$

then the m^{th} NAL unit is regarded as a candidate. Note that the inequality is between the $(m-1)^{\text{th}}$ and m^{th} NAL units. The score of this candidate, S_m, is defined as the number of instances among $k \in \{[2,7] \cup [14,16] \cup 18\}$, (6) is satisfied.

Next the implications of *first_slice_segment_in_pic_flag* are incorporated into S_m. The bit pertaining to *first_slice_segment_in_pic_flag* of the i^{th} NAL unit is $\hat{u}[j_{i-1}+17]$. If $\hat{u}[j_{m-1}+17] = 1$, then S_m is incremented by 1.

The candidates are shortlisted as $\mathbf{A} = \{m | S_m \geq 3\}$. The value 3 has been empirically chosen such that all actual first NAL units are included in the candidate set $\mathbf{A}$. Fig. 2 depicts experimental results on successful detection and false detection of first NAL units, for various S_m thresholds. The experiment is carried out on the *Soccer* sequence, using the simulation parameters indicated in Table 3, at a channel SNR of 11.5dB. This is the lowest SNR value, at which the actual first NAL units can be identified using the notion of an S_m threshold.

In Fig. 2, the bars in black ink indicate the actual first NAL units residing in $\mathbf{A}$, as a percentage of all actual first NAL units in the transport block. The highest threshold value such that all actual first NAL units are included in $\mathbf{A}$ is 3. The gray bars indicate the false first NAL units residing in $\mathbf{A}$, as a percentage of all NAL units included in $\mathbf{A}$. Note that at the threshold of '3' there exists some false positives residing in $\mathbf{A}$.

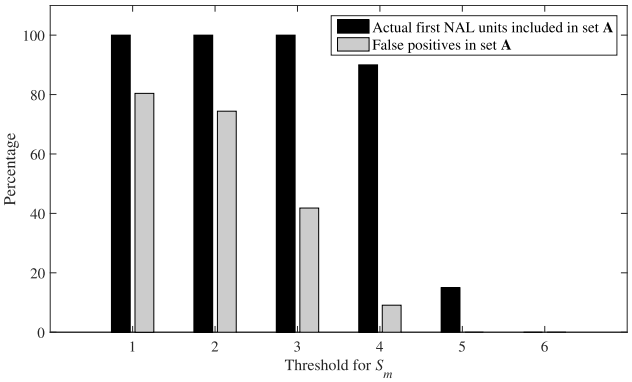


FIGURE 2. Validity of A for various S_m thresholds at 11.5dB.

Algorithm 2 Final Screening Criteria for $\mathbf{A}$

Require: $\hat{\mathbf{C}}_t$, $\mathbf{A}$
Ensure: false first NAL units are eliminated from $\mathbf{A}$
  repeat
    $flag \leftarrow 0$
    for $m \in [i, i+n]$ **do**
      if $m \in \mathbf{A}$ **then**
        $\hat{u}[j_{m-1}+17] \leftarrow 1$ ▷ *first_slice_segment_in_pic_flag*
      else
        $\hat{u}[j_{m-1}+17] \leftarrow 0$
      end if
    end for
    for $m \in \mathbf{A}$ **do** ▷ every access unit
      $R_m \leftarrow 0$
      obtain $\{\mathbf{E}_k\}$ ▷ includes 22 sequences
      for each sequence, $\mathbf{E}_k$, **do**
        if mode≠first member **then**
          $R_m \leftarrow R_m + 1$
        end if
      end for
      if $R_m \geq 8$ **then**
        $\mathbf{A} \leftarrow \mathbf{A} - \{m\}$
        $flag \leftarrow 1$
      end if
    end for
  until $flag \wedge (\mathbf{A} \neq \emptyset)$

Nevertheless, we proceed on the hypothesis that the elements in $\mathbf{A}$ are legitimate, and modify the corresponding *first_slice_segment_in_pic_flag* fields. This is because the value of this flag changes the course of how the subsequent fields are parsed.

The following assignment is performed to modify the *first_slice_segment_in_pic_flag*:

$$\hat{u}[j_{m-1}+17] = \begin{cases} 1, & \text{if } m \in \mathbf{A} \\ 0, & \text{otherwise} \end{cases} \quad (7)$$

A final screening criteria for $\mathbf{A}$ is introduced to eliminate the inaccurate candidates in $\mathbf{A}$. The relevant algorithm is illustrated below and summarized in Algorithm 2. We use the

concept that certain fields must hold common values for all NAL units within an access unit. The considered fields are those indicated in Table 1 and the following fields (includes all bits of *slice_pic_parameter_set_id*).

- *slice_type*
- *slice_pic_order_cnt_lsb*
- *short_term_ref_pic_set_sps_flag*

For each of the selected 6 fields, the collocated bit positions of the same field in different NAL units that fall within an access unit are considered. For example, when considering the first bit position of *slice_pic_order_cnt_lsb*, the considered[2] bit sequence is, $\mathbf{E} = \{\hat{u}[j_{m-1}+22], \hat{u}[j_m+27], \hat{u}[j_{m+1}+27], ..., \hat{u}[j_{\acute{m}-2}+27]\}$, where m and $\acute{m}$ are two adjacent members in $\mathbf{A}$.

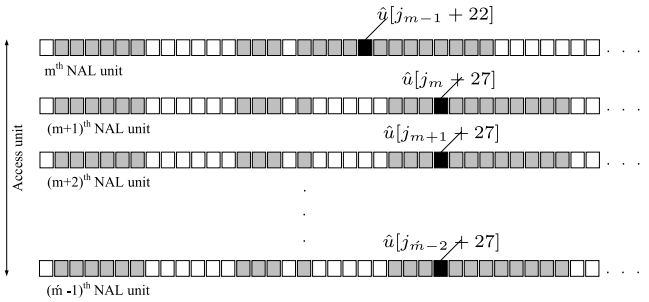


FIGURE 3. Bit positions considered in the final screening process of A.

The bit positions of $\mathbf{E}$ are indicated in Fig. 3 as black. Gray indicates other bit positions that are considered. Note that sequence $\mathbf{E}$ is one of the twenty two different bit sequences that need to be analyzed (the summation of the lengths of the six fields is 22). The twenty two different bit sequences are denoted as $\{\mathbf{E}_k\}$ in Algorithm 2.

It should be noted that the *first_slice_segment_in_pic_flag* changes the course of how the subsequent fields are read [24]. If an invalid candidate resides in $\mathbf{A}$, the fields of this invalid candidate NAL unit must be different from the corresponding fields of the other NAL units within the access unit. Therefore, the number of instances the bits of the first NAL unit ($\hat{u}[j_{m-1}+22]$ in the example) differs from each bit's respective mode (mode of $\mathbf{E}$ in the example) is counted, and denoted as R_m.

If $R_m \geq 8$, m is removed from $\mathbf{A}$. The value '8' is empirically chosen such that the actual first NAL units are not eliminated from $\mathbf{A}$, yet a majority of false positives are eliminated. The results in Fig. 4 are derived with similar simulation parameters as those used in the S_m threshold identification experiment. In Fig. 4, the bars in black ink indicate the actual first NAL units eliminated from $\mathbf{A}$, as a percentage of all actual first NAL units residing in $\mathbf{A}$. The lowest threshold value such that none of the actual first NAL units are eliminated is 8. The bars in gray ink indicate the false positives eliminated from $\mathbf{A}$, as a percentage of all false positives included in $\mathbf{A}$. Note that at the threshold of '8', Algorithm 2 does not succeed in eliminating all false first NAL units.

[2]Here it is assumed that $b \leq 16$ or $23 \leq b$; $\mathbf{f}$ is 1-bit long; $\mathbf{g}$ is 5-bit long; and $\mathbf{h}$ is 3-bit long. In Picture Parameter Set (PPS), *dependent_slice_segment_flag* = 0, *num_extra_slice_header_bits* = 0, *output_flag_present_flag* = 0, and in Sequence Parameter Set (SPS), *separate_colour_plane_flag* = 0, *long_term_ref_pics_present_flag* = 0. b, f, g and h are fields in Table 2

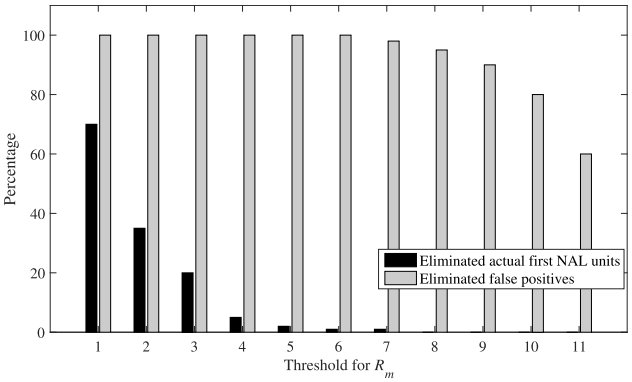


FIGURE 4. Validity of A for various R_m thresholds.

Based on this new $\mathbf{A}$, (7) is re-performed until, for all candidates in $\mathbf{A}$, the threshold R_m is not surpassed. Subsequently, field modification is initiated. The very rare occasion when a false positive survives Algorithm 2, is tackled in Algorithm 4.

V. HEADER FIELD CORRECTION

The field modifications described below occur after the access unit boundary identification, and occur at the SPC module in Fig. 1. The validation rules are categorized to 4 types as illustrated in the Table 2. The character tag (**a**,**b**,**c**) given to each field is based on the order the fields are read at the decoder, as per Section 7.3.6.1 of [24]. Reading and correcting the header fields are performed in the same order, one access unit at a time. The algorithm used to correct the field is based on the category to which the field belongs in Table 2.

A. CATEGORY (a)

The fields in Category (a) have a definite value and therefore, the relevant bits are modified as per the rules in Table 2 to either 0 or 1. Thereafter, the corresponding LLR values are modified. Since the alterations in Category (a) are highly reliable decisions the following modification is performed on the relevant bit:

$$L'(u[k]) = \begin{cases} -r, & \text{if } \hat{u}[k] = 0 \\ +r, & \text{if } \hat{u}[k] = 1 \end{cases} \quad (8)$$

The value $r > 0$ is determined through empirical data, and $L'(u[k])$ is the modified value of the member, $L(u[k])$, in $\mathbf{C}_t$. For example, if $\mathbf{e}$ (refer Table 2) in the i^{th} NAL unit is modified to 0, the following assignment is done: $L'(u[j_{i-1}+17]) = -r$.

B. CATEGORY (b)

For each field in Category (b), the collocated bit positions of the same field in different NAL units that fall within an

TABLE 2. Header fields.

Field and Syntax	Rules
Category (a)	
a forbidden_zero_bit f(1) **c** nuh_layer_id u(6) **e** first_slice_segment_in_pic_flag u(1)	**a** is always zero. It is assumed in this paper that scalable video coding is not used, hence **c** is always zero. **e** is '1' only at the first NAL unit of an access unit, else it is zero.
Category (b)	
b nal_unit_type u(6) **d** nuh_temporal_id_plus1 u(3) **f** slice_pic_parameter_set_id ue(v) **h** slice_type ue(v) **i** slice_pic_order_cnt_lsb u(v) **j** short_term_ref_pic_set_sps_flag u(1) **k** slice_temporal_mvp_enabled_flag u(1) **o** slice_qp_delta se(v)	These fields hold common values within an access unit.
Category (c)	
g slice_segment_address u(v) **l** num_ref_idx_l0_active_minus1 ue(v) **m** collocated_ref_idx ue(v) **n** five_minus_max_num_merge_cand ue(v)	**g** is ascending within the access unit. **l** is in the range of 0 to 14, inclusive. **m** is in the range of 0 to **l**, inclusive. **n** is in the range of 0 to 4, inclusive.
Category (d)	
p byte_alignment	Must be a power of two.
Category (e)	
b, **d**, **h**, **i**, **o**; **b**, **d** and **h** are related to its neighboring access units and the GoP size, in that they repeat between GoPs. **d**=1 for intra pictures (i.e., $16 \leq \mathbf{b} \leq 21$). **i** takes different values between access units. **o** is always positive and is in the neighborhood of the previous access unit's **o** value.	

In syntax notations; f: fixed, u: unsigned, ue: Exp-Golomb, se: signed Exp-Golomb. The value inside the braces indicate the bit length. The bit lengths of **i** and **g** are derived from other fields in the SPS.

access unit are considered. Denote these bit sequences as $\{\mathbf{E}_k\}$, the function f_a is performed on each $\mathbf{E}_k$ to obtain $\left\{\hat{\mathbf{E}}_k\right\}$. Identification of $\{\mathbf{E}_k\}$ is performed in a similar manner as explained in Fig. 3.

Some fields in Category (b) are variable length coded (Exp-Golomb or signed Exp-Golomb), e.g., **f**, **h** and **o**. Exp-Golomb and signed Exp-Golomb syntaxes have a well defined structure (please refer [24, Sec. 9]). Their representation is pivotable about a center '1' bit. The number of zero bits to precede the '1' bit, is the same number of bits that follow the '1' bit. This characteristic is used when correcting **f**, **h** and **o**. The algorithm to obtain the collocated bit sequences for these fields is explained in Algorithm 3.

Since most fields in Category (b) relate to other access units (fields which exhibit such relationships are categorized into Category (e)), the values extrapolated from the previous access units are deployed to verify the authenticity of Category (b) fields. This procedure is explained in Algorithm 4.

The condition, '*if* $3n_{\bar{\delta}} > \acute{m} - m$ and $2n_{\bar{\delta}} > 3n_{\bar{\bar{\delta}}}$', ensures that $\bar{\delta}$ occupies at least a third of the fields within the access unit, and that it is at least 3/2 times as popular as the next popular candidate. These popularity conditions are chosen based on the analysis of compressed and corrupted HEVC sequences. If the condition is not satisfied, it implies that our choice of first NAL unit positions (**A**) is inaccurate. Consequently, none of the headers in this access unit are modified, and are reset to their original LLR values.

Algorithm 3 Obtaining Collocated Bit Sequences and Eliminating Sparse Bit Errors For Category (b) Fields

Require: access units are segmented as per the first NAL units in **A**

Ensure: collocated bit sequences to be used in Algorithm 4

```
for all fields in Category (b) do
    if fixed length field then
        obtain collocated bit sequences of the corresponding bit positions and perform f_a on each sequence
    else                                   ▷ Exp-Golomb or signed Exp-Golomb
        obtain collocated bit sequences of the first bit position and perform f_a
        b ← 0
        while mode of last obtained sequence = 0 do
            obtain collocated bit sequences of the next bit position and perform f_a
            b ← b + 1
        end while
        while b > 0 do
            obtain collocated bit sequences of the next bit position and perform f_a
            b ← b − 1
        end while
    end if
end for
```

In Algorithm 4, the operations 'obtain field value, δ_i' and 'modify LLR stream' require further explanation. For fixed length fields, these operations are, respectively, a binary to decimal conversion, and performing the modification (8) on defined LLR positions. However, for Exp-Golomb and signed Exp-Golomb syntaxes the methodology is different.

Denote the collocated bit sequence for the first bit of such a field as $\left\{\hat{u}[j_{i-1}+k_i] \mid i \in [m, \acute{m}-1]\right\}$. Note that these values relate to the m^{th} NAL unit through to the $(\acute{m}-1)^{\text{th}}$ NAL unit. k_i are the indices on which the field starts, with respect to the beginning of each NAL unit. For Exp-Golomb syntaxes, the field value is read as:

$$\delta_i = \sum_{x=1}^{b} \left(2^{x-1} + 2^{b-x}(\hat{u}[j_{i-1}+k_i+b+x])\right) \tag{9}$$

where b is constrained by $\hat{u}[j_{i-1}+k_i+b]=1$ and $\forall x \in [0, b-1], \hat{u}[j_{i-1}+k_i+x]=0$. For signed Exp-Golomb syntaxes, the field value is read as:

$$\delta_i = (-1)^{a+1}\lceil a/2 \rceil \tag{10}$$

Algorithm 4 Category (b)

Require: availability of historic slice headers, bit values of an access unit

Ensure: corrected header fields or relinquishment of the correction attempt

```
for each field in Category (b) do
    obtain collocated bit sequences                ▷ Algorithm 3
    obtain set of possible extrapolated values Ω
    for each NAL unit do                           ▷ i ∈ [m, ḿ − 1]
        obtain field value, δ_i
        if δ_i ∉ Ω then
            δ_i ← 0
        end if
    end for
    δ̄ ← most popular non-zero δ_i value
    n_δ̄ ← its frequency
    δ̿ ← next most popular non-zero δ_i value
    n_δ̿ ← its frequency
    if 3n_δ̄ > ḿ − m and 2n_δ̄ > 3n_δ̿ then
        corrected header field value ← δ̄
        modify LLR stream
    else
        reset all LLR values of this access unit
        return
    end if
end for
```

where a is the value when read as an Exp-Golomb syntax field (9), and $\lceil\cdot\rceil$ denotes the closest integer larger than its argument. Equations (9) and (10) describe the 'obtain field value, δ_i' operation.

When the corrected header field value for Exp Golomb syntaxes is selected as $\bar{\delta}$, the 'modify LLR stream' operation is performed as follows:

$$L'(u[n_i + x]) = \begin{cases} -r, & \text{if } x \in [0, \beta - 1] \\ +r, & \text{if } x = \beta \\ r(2\lambda_x - 1) & \text{if } x \in [\beta + 1, 2\beta] \end{cases} \quad (11)$$

where n_i is the index on which the field starts for the i^{th} NAL unit (i.e., $n_i = j_{i-1} + k_i$)
$\beta = \arg\max_{b \in \mathbb{Z}^+} \left\{ \sum_{x=1}^{b} 2^{x-1} \leq \bar{\delta} \right\}$, and
$(\lambda_{\beta+1} \ldots \lambda_{2\beta})$ is the binary representation of $(\bar{\delta} - \sum_{x=1}^{\beta} 2^{x-1})$.

For **o**, which is the only signed Exp-Golomb syntax field in the slice header (note that this field is always positive), when the corrected header field value is selected as $\bar{\delta}_s$, it is first converted to obtain $\bar{\delta} = 2\bar{\delta}_s - 1$, which is used to perform the modification in (11).

C. CATEGORY (c)

Errors in Category (c) are only detected and cannot be corrected. When a non compliance is identified, the reliability of the relevant error detected bits is reduced. This modification is confined to one NAL unit. In such instances where an error is detected in a header field, yet no reliable conclusion on its bit values can be decided, the following modification is performed:

$$L'(u[k]) = \begin{cases} \ln\{\exp(L(u[k])) \cdot s\}, & \text{if } \hat{u}[k] = 0 \\ \ln\{\exp(L(u[k]))/s\}, & \text{if } \hat{u}[k] = 1 \end{cases} \quad (12)$$

Here, $\hat{u}[k]$ is a bit in the erroneous field, and the weighting factor $s > 1$ is selected such that the magnitude of $L(u[k])$ is reduced, but the hard decision remains unaltered.

D. CATEGORY (d)

The only field in Category (d), **p**, is corrected as follows. The slice header has an integer number of bytes. In order to achieve this, the compression function pads the header with a '1' bit followed by '0' bits until the header length is an integer number of bytes. Therefore, after all the fields in the header have been read, we alter the LLR sequence to indicate a '1' followed by '0' bits until the byte is complete. The LLR modification is as per (8).

E. CATEGORY (e)

Category (e) fields exhibit relationships with other fields, and therefore, the value of these fields can be predicted. Extrapolated values for these fields are used as Ω in Algorithm 4. Obtaining candidates for the extrapolated values is done by identifying a pattern among the field values of the past access units. A few examples are described in Table 2 under Category (e). For the remaining fields in Category (b) (i.e., the Category (b) fields absent from Category (e)), the set of possible extrapolated values (Ω in Algorithm 4) is the set of all integers.

VI. APPLICATION TO REALISTIC VIDEO SEQUENCES

It is known that given a k-bit compressed video sequence, not all 2^k bit patterns result in valid source sequences. This indicates unexploited bit stream redundancy. The algorithms proposed, in effect, utilize this noncompliance for correcting some transmission errors. By doing so, we implement JSCD. This section aims to demonstrate how to apply the information theoretic algorithms to a practical system scenario, and evaluate the performance gain achieved at the turbo decoder in terms of error correction capability.

To this end, an exemplary LTE-A system model, which transmits over an Extended Pedestrian A channel suggested by International Telecommunication Union, is used in simulations. The raw video data are compressed using the HEVC reference codec [24], with the reference implementation publicly available at [29]. The configuration file is provided at the same repository, *encoder_lowdelay_P_main.cfg* with modifications to restrict the maximum allowed NAL unit size, and to address the video resolution and frame rate. After video compression, 40 bytes are allocated to precede each NAL unit to account for the combined overhead of IP/UDP/RTP headers [25]. A maximum IP packet size of 100 bytes is adopted as assumed by JVT's wireless common

conditions [25]. Simulations are performed using the MATLAB LTE-System Toolbox, which has been tested and validated [30].

TABLE 3. Simulation setup.

CHANNEL PARAMETERS	
Input signal sample rate	15.36MHz
EPA path delay vector (ns)	[0 30 70 90 110 190 410]
EPA path gain vector (dB)	[0 -1 -2 -3 -8 -17.2 -20.8]
Number of transmit/receive antennas	2×2
There is no spatial correlation between the transmit and receive antennas	
Fading Distribution	Rayleigh
Fading technique used to model the channel	Filtered Gaussian noise
Channel bandwidth	10MHz
VIDEO CODING PARAMETERS	
Frame rates for the three videos; *Foreman*, *Soccer* and *Beergarden*	30, 60, 60
Number of frames	288
Video resolutions	352×288, 704×576, 1920×1080
Coding tree unit size	64×64
Group of Picture (GoP) size	4 frames
Intra period	32 frames
Quantization parameter	32
Maximum number of bytes per slice	100
CHANNEL CODING PARAMETERS	
Code rate	0.5
Modulation type	16QAM
Maximum number of turbo iterations	8
CRC length	24 bits

A. ERROR PERFORMANCE OF THE JOINT SOURCE CHANNEL DECODING ALGORITHM

Experiments have been carried out to verify the benefits of the SPC module. The simulation parameters for video compression, channel encoding and transmission are as summarized in Table 3. Two instances of turbo decoding are considered with two different LLR sequences being input to the turbo decoder: one sequence is the unmodified, received LLR sequence (indicated by $\mathbf{y}$ in Fig. 1), and for the other instance, a modified version of $\mathbf{y}$. In this modification, the systematic bit LLRs of $\mathbf{y}$, $\left\{y_k^s \mid k \in [(t-1)T+1, tT]\right\}$ are regarded as $\mathbf{C}_t$, and passed through the SPC module and LLR modification module, before being input to the turbo decoder. Performance of the two turbo decoding instances is evaluated by verifying the integrity of the transport blocks after each turbo iteration, using the CRC.

In this experiment, the HEVC compressed sequence of *Foreman* has been used. A total of 60 transport blocks from this video sequence are considered, and for each transport block transmission, the number of iterations executed until the CRC is successful is recorded. Fig. 5 depicts excerpts from these results. Note that the y axis indicates the cumulative number of successful transport blocks after each iteration number. No transport block recoveries were observed beyond 15 turbo iterations.

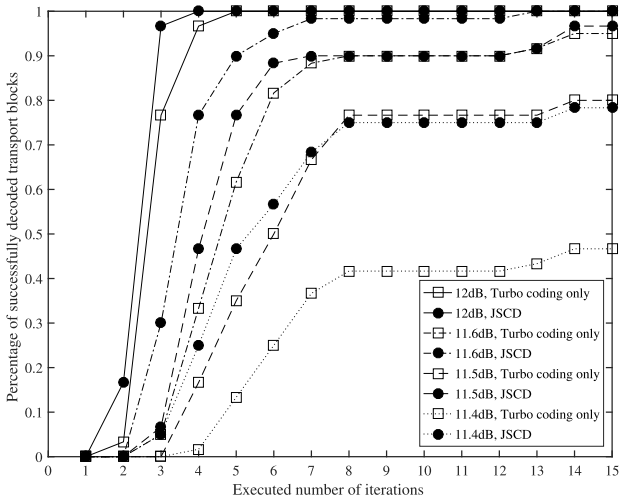


FIGURE 5. Contribution from the SPC module.

The recorded results and Fig. 5 indicate that the SPC module contributes towards the turbo decoding performance at all channel SNR levels above 11.3dB, below which the LLR stream becomes irreparable. As per the recorded results, 11.7dB is the minimum channel SNR required to recover all transport blocks when turbo decoding is used in isolation (This is not indicated in Fig. 5). The inclusion of the SPC module has reduced this threshold to 11.6dB.

Furthermore, the achieved gain increases as the channel deteriorates. For example, at an SNR level of 11.5dB in the absence of the SPC module, 20% of the transport blocks cannot be recovered. Inclusion of the SPC module reduces this value to 3% indicating better chances of successfully concealing the video artifacts due to lost NAL units. In comparison, at 11.4dB, these values stand at 53% and 22%, respectively. In other words, at 11.5dB the recovery increased by 17% and at 11.4dB recovery increased by 31%.

Fig. 5 also indicates the effectiveness of the introduced interleaver, π, in recovering the entire video stream, not only the header regions. If the recovery was only in the header regions, and the majority of the transport block remained unaltered, hardly any improvement can be observed. Due to the interleaving procedure, the modified header LLR values reside among the CABAC coded slice data and assist in their recovery.

B. CHOICE OF MAXIMUM TURBO DECODER ITERATIONS

Another feature evident in Fig. 5 is that no transport blocks are recovered between turbo iterations 8 and 12. This feature can be explained from EXIT charts, and it can assist in the decision to choose the maximum number of turbo iterations in the proposed algorithm. The EXIT charts for the turbo code used in the simulations are depicted in Fig. 6. The curves

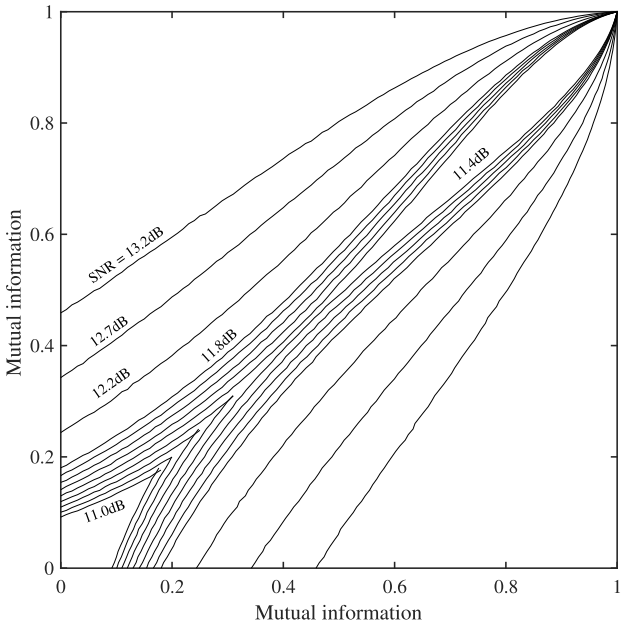


FIGURE 6. EXIT chart for the LTE-Advanced turbo decoder, when used in the simulation setup in this paper. The two axes indicate the mutual information between the transmitted bits and the extrinsic LLR values conveyed to the counterpart convolutional decoder, and the mutual information between the transmitted bits and a priori LLR values input to the convolutional decoder.

between SNR values of 11.0dB and 11.8dB are in steps of 0.1dB. Note that the curves are presented only up to their first intersection.

Each transport block transmission in the above experiment gives rise to a unique manifestation of an EXIT chart. The EXIT charts depicted in Fig. 6 are averaged graphs of these manifestations. For example, if we consider the '11.6dB, Turbo coding only' graph in Fig. 5, 95% of the transport block transmissions have resulted in EXIT charts which allow the trajectory to 'exit'. The remaining 5% of the manifestations have resulted in blocked bottlenecks and therefore, the trajectory cannot exit.

The number of executed iterations increases as the bottleneck becomes narrower. If the bottleneck becomes too narrow, the required number of executed iterations to sneak through the bottleneck drastically increases. This is the reason for the absence of recovered transport blocks in Fig. 5 between turbo iterations 8 and 12. A transport block that succeeded in exiting within 8 iterations, implies EXIT chart manifestations with loose bottlenecks. If the transport block encountered a tight bottleneck, it cannot exit until after 12 or 13 iterations.

Thus, a maximum of 8 turbo iterations is chosen for the proposed algorithm. After 8 iterations, the majority of recoverable transport blocks can be recovered. The remaining transport blocks (if any can be recovered) can only be recovered after many more turbo iterations. There is also a very high probability that these cannot be recovered at all in this turbo decoding stage. Therefore, the intelligent choice is to temporarily seize turbo decoding after the 8^{th} iteration and recommence turbo decoding after LLR modification.

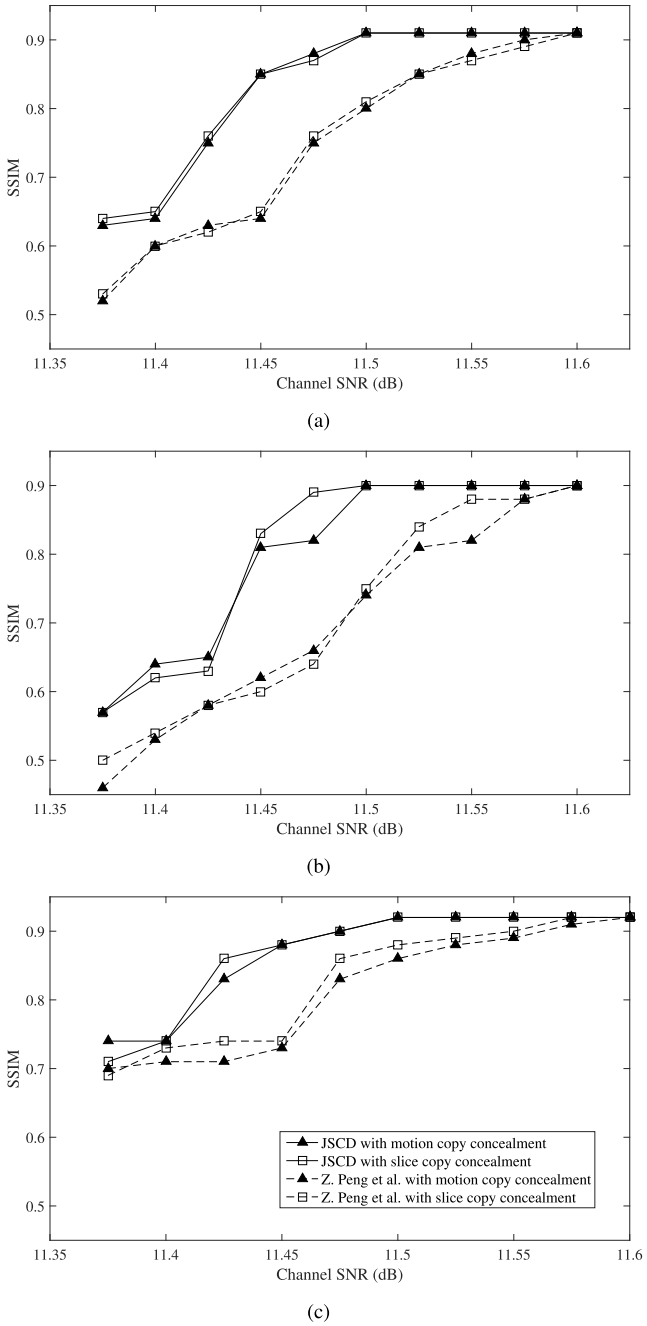


FIGURE 7. Effects on user viewing experience. (a) Foreman: 352 × 288. (b) Soccer: 704 × 576. (c) Beergarden: 1920 × 1080.

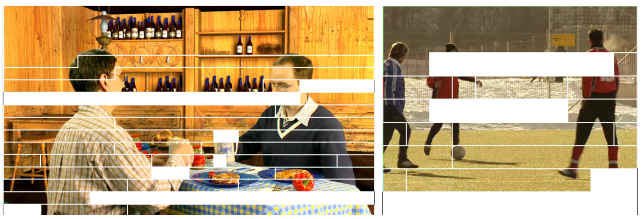
C. IMPROVEMENT IN END USER VIEWING EXPERIENCE

Fig. 7 compares the end-to-end performance of the proposed algorithm in terms of objective video quality of the decoded video sequences. Results for three video resolutions have been demonstrated. The simulation setup is as per Table 3. The objective quality of the videos is measured using the metric Structural Similarity (SSIM) index [31], which is a full reference metric.

In this experiment, the maximum number of LLR modification stage executions (refer Fig. 1) per transport block is limited to three. If the recovery is incomplete after three cycles, the relevant NAL units within the transport block are considered to be lost. At the video decompression stage, when a slice is lost, the affected regions are recovered from error concealment. Results are presented for two forms of error concealment algorithms; slice copy concealment where the previous frame's relevant region is copied in lieu of the lost slice, and motion copy concealment [32] where the motion vector of the previous frame's relevant pixel block is regarded as the lost block's motion vector to synthesize the lost regions.

As the benchmark for this experiment, the JSCD algorithm in [20] for MPEG video sequences has been adapted. Reference [20] modified the turbo decoder's extrinsic information pertinent to MPEG picture start codes and MPEG slice start codes. It was developed under the assumption that the positions of these start codes have been previously recovered by some robust scheme. Since the HEVC equivalent for these start codes are the NAL unit delimiters, this experiment includes the delimiters in its packetized transmission. The benchmark modifies the LLRs of these delimiter bit positions using the '*EFSIF*' algorithm introduced in [20], while the JSCD method modifies both the delimiter positions and slice header positions using the algorithms described in this paper.

The results indicate that the proposed method outperforms the benchmark method. Fig. 7 depicts SSIM measurements only above 11.375dB, below which hardly any NAL unit is recoverable, and therefore measurement of SSIM is irrelevant. An interesting feature to note is that the LLR modification of header fields causes a gain of 0.1dB in channel SNR over the delimiter modification approach presented in [20]. The reason for this gain is because the method proposed in this paper aims to modify more LLR positions than being confined to delimiters.



(a) (b)

FIGURE 8. Lost slices at different video resolutions. (a) *Beergarden* frame: 1920 × 1080. (b) *Soccer* frame: 704 × 576.

The results show that *Beergarden* video has better resilience to channel degradation over lower resolution videos. Keeping in mind that NAL unit size is restricted to 100 bytes, a high definition (HD) frame entails more slices than a lower resolution frame does, as indicated in Fig. 8. In the event of a slice loss in a low resolution video, a vast area is lost. At a given channel SNR level, although the slice loss rate is equal for both videos shown in Fig. 8, the areas lost in the HD frame are scattered across the frame, thus making the error concealment more effective.

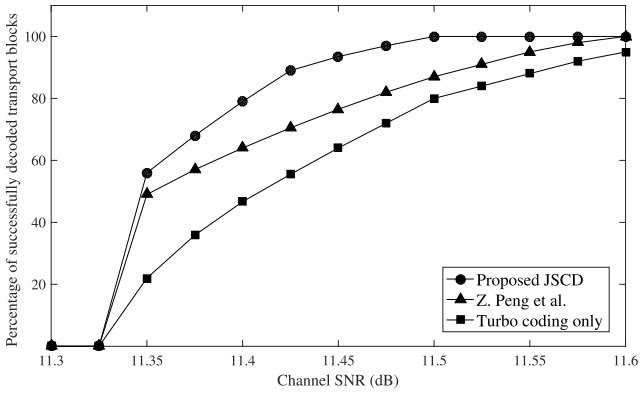


FIGURE 9. Overall payload recovery performance of the proposed methodology.

The performance of the proposed methodology is further demonstrated in a generic context. Fig. 9 depicts the percentage of successfully decoded transport blocks in the previous experiment after a maximum of three iterations of LLR modification. The percentage value is the weighted average of the three video sequence outcomes.

It is evident that below 11.35dB, the recovered amount of transport blocks from the turbo decoding iteration is not sufficient for the proposed methodology to operate. The proposed methodology exhibits a maximum additional transport block saving gain of 33.5% at 11.425dB, as opposed to the turbo decoding only method. Against the benchmark method, this value records as 18.5%

The innovation proposed is in line with the carbon foot print reducing initiative embraced by many service providers and next generation telecommunication standardization committees. The gain of 0.1dB represented in Fig. 7 transforms to a power saving of 2.28% at the transmitting base station.

VII. CONCLUSION

This paper presents a cross layer algorithm to exploit HEVC source redundancy in collaboration with channel code redundancy for error recovery in mobile video transmission. The exploited source redundancy exist in the form of slice header semantics and field patterns. As a prerequisite to identifying the selected HEVC fields, the paper presents a boundary identification algorithm for access units in a corrupted bit sequence. Subsequently, algorithms are developed to deduce the possible values of the selected fields. The deduced values are used to modify LLR values, that are subsequently used as extrinsic information in the turbo decoder.

Experimental results demonstrate that the proposed methodology has a favorable effect on visual experience. The minimum channel SNR required to recover all transport blocks is reduced from 11.7dB to 11.6dB. This represents a power saving of 2.28% at the transmitter base station. Experiments also revealed that the proposed algorithms can achieve improvements at all SNR values above 11.3dB.

Despite being a JSCD approach, the proposed technology does not require the application layer decompression function. It relies only on computationally inexpensive semantic conformance and trend identification functions, which utilize redundancies present in slice headers. However, potential redundancies in slice data can further enhance the error correction capabilities. Although the CABAC attribute makes this extremely complex, its ultimate achievement will pay off with huge benefits. Therefore, analyzing slice data for improving the accuracy of LLR values is proposed as a future work.

REFERENCES

[1] Qualcomm Technologies Inc. (2014). *Enabling the Full 4k Mobile Experience: System Leadership*, accessed on Nov. 19, 2015. [Online]. Available: https://www.qualcomm.com/media/documents/files/enabling-the-full-4k-mobile-experience-system-leadership.pdf

[2] (2016). *Cisco Visual Networking Index: Global Mobile Data Traffic Forecast Update, 2015Ű2020 White Paper*, accessed on Aug. 17, 2016. [Online]. Available: http://www.cisco.com/en/US/solutions/collateral/ns341/ns525/ns537/ns705/ns827/white_paper_c11-520862.html

[3] G. J. Sullivan, J.-R. Ohm, W.-J. Han, and T. Wiegand, "Overview of the high efficiency video coding (HEVC) standard," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 22, no. 12, pp. 1649–1668, Dec. 2012.

[4] R. Perera, A. Fernando, T. Mallikarachchi, H. K. Arachchi, and M. Pourazad, "QoE aware resource allocation for video communications over LTE based mobile networks," in *Proc. 10th Int. Conf. Heterogeneous Netw. Quality Rel. Secur. Robustness (QShine)*, Aug. 2014, pp. 63–69.

[5] M. Hafeez, S. Jangsher, and S. A. Khayam, "A cross-layer architecture for motion-adaptive video transmission over MIMO channels," in *Proc. IEEE Int. Conf. Commun.(ICC)*, Jun. 2011, pp. 1–5.

[6] R. Perera, A. Fernando, H. K. Arachchi, and M. A. Imran, "Adaptive modulation and coding based error resilience for transmission of compressed video," in *Proc. Int. Wireless Commun. Mobile Comput. Conf. (IWCMC)*, Aug. 2015, pp. 1127–1132.

[7] B. Sklar, *Fundamentals of Turbo Codes*. Englewood Cliffs, NJ, USA: Prentice-Hall, 2002.

[8] H. Nguyen and P. Duhamel, "Estimation of redundancy in compressed image and video data for joint source- channel decoding," in *Proc. IEEE Global Telecommun. Conf.*, vol. 4. Dec. 2003, pp. 2198–2202.

[9] D. Marpe, H. Schwarz, and T. Wiegand, "Context-based adaptive binary arithmetic coding in the H.264/AVC video compression standard," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 13, no. 7, pp. 620–636, Jul. 2003.

[10] *Video Coding for Low Bit Rate Communication*, document H.263, International Telecommunication Union, 2005.

[11] H. Nguyen and P. Duhamel, "Iterative joint source-channel decoding of VLC exploiting source semantics over realistic radio-mobile channels," *IEEE Trans. Commun.*, vol. 57, no. 6, pp. 1701–1711, Jun. 2009.

[12] G. D. Forney, Jr., "The Viterbi algorithm," *Proc. IEEE*, vol. 61, no. 3, pp. 268–278, Mar. 1973.

[13] R. A. Farrugia and C. J. Debono, "A hybrid error control and artifact detection mechanism for robust decoding of H.264/AVC video sequences," *IEEE Trans. Circuits Syst. for Video Technol.*, vol. 20, no. 5, pp. 756–762, May 2010.

[14] R. Farrugia and C. Debono, "Robust decoder-based error control strategy for recovery of H.264/AVC video content," *IET Commun.*, vol. 5, no. 13, pp. 1928–1938, Sep. 2011.

[15] N. Q. Nguyen, W. E. Lynch, and T. Le-Ngoc, "Iterative joint source-channel decoding for H.264 video transmission using virtual checking method at source decoder," in *Proc. CCECE*. May 2010, pp. 1–4.

[16] G. Sabeva, S. Jamaa, M. Kieffer, and P. Duhamel, "Robust decoding of H.264 encoded video transmitted over wireless channels," in *Proc. IEEE Workshop Multimedia Signal Process.*, Oct. 2006, pp. 9–13.

[17] X. F. Ma and W. Lynch, "Iterative joint source-channel decoding using turbo codes for MPEG-4 video transmission," in *Proc. IEEE Int. Conf. Acoust., Speech, Signal Process.*, vol. 4. May 2004, pp. iv-657–iv-660.

[18] D. Levine, W. E. Lynch, and T. Le-Ngoc, "Iterative joint source-channel decoding of H.264 compressed video," in *Proc. IEEE Int. Symp. Circuits Syst.*, May 2007, pp. 1517–1520.

[19] F. Caron and S. Coulombe, "Video error correction using soft-output and hard-output maximum likelihood decoding applied to an H.264 baseline profile," *IEEE Trans. Circuits Syst. for Video Technol.*, vol. 25, no. 7, pp. 1161–1174, Jul. 2015.

[20] Z. Peng, Y.-F. Huang, and D. J. Costello, "Turbo codes for image transmission-a joint channel and source decoding approach," *IEEE J. Sel. Areas Commun.*, vol. 18, no. 6, pp. 868–879, Jun. 2000.

[21] P. C. Cosman, K. L. Oehler, E. A. Riskin, and R. M. Gray, "Using vector quantization for image processing," *Proc. IEEE*, vol. 81, no. 9, pp. 1326–1341, Sep. 1993.

[22] J. Hagenauer, "Source-controlled channel decoding," *IEEE Trans. Commun.*, vol. 43, no. 9, pp. 2449–2457, Sep. 1995.

[23] T. Wiegand, G. J. Sullivan, G. Bjøntegaard, and A. Luthra, "Overview of the H.264/AVC video coding standard," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 13, no. 7, pp. 560–576, Jul. 2003.

[24] *High efficiency video coding*, ITU-T document Rec. H.265, International Telecommunication Union, 2013.

[25] S. Wenger, "H.264/AVC over IP," *IEEE Trans. Circuits Syst. for Video Technol.*, vol. 13, no. 7, pp. 645–656, Jul. 2003.

[26] E. Dahlman, S. Parkval, and J. Skold, *4G: LTE/LTE-Advanced for Mobile Broadband*. New York, Ny, USA: Elsevier, 2011.

[27] "ITU-T Recommendation H.265: High efficiency video coding," Int. Telecommun. Union, Switzerland, Geneva, Tech. Rep. H.265 (04/2013), 2013.

[28] S. ten Brink, "Convergence behavior of iteratively decoded parallel concatenated codes," *IEEE Trans. Commun.*, vol. 49, no. 10, pp. 1727–1737, Oct. 2001.

[29] Fraunhofer. (2013). *HEVC Reference Software Manual*. [Online]. Available: https://hevc.hhi.fraunhofer.de/

[30] MathWorks, *LTE System Toolbox*, accessed on May 26, 2016. [Online]. Available: http://uk.mathworks.com/products/lte-system/features.html

[31] Z. Wang, A. C. Bovik, H. R. Sheikh, and E. P. Simoncelli, "Image quality assessment: From error visibility to structural similarity," *IEEE Trans. Image Process.*, vol. 13, no. 4, pp. 600–612, Apr. 2004.

[32] G. Kulupana, D. S. Talagala, H. K. Arachchi, and A. Fernando, "Error resilience aware motion estimation and mode selection for HEVC video transmission," in *Proc. IEEE Int. Conf. Consum. Electron.*, Jan. 2016, pp. 85–86.

RYAN PERERA received the B.Sc. Eng. degree (Hons.) in electronic and telecommunication engineering from the University of Moratuwa, Sri Lanka, in 2013. He is currently pursuing the Ph.D. degree in electronic engineering with the University of Surrey, U.K. In 2013, he was a Business Analyst with Millennium IT, an affiliation of London Stock Exchange Group. His current research interests include channel adaptive error resilience techniques for real time video communication, optimization of LTE-Advanced physical layer communication, and cloud computing techniques to reinforce mobile communications.

HEMANTHA KODIKARA ARACHCHI (M'02) received the B.Sc. Eng. degree (Hons.) and the M.Phil. degree in electronic and telecommunication engineering from the University of Moratuwa, Sri Lanka, in 1997 and 2000, respectively, and the Ph.D. degree in telecommunications from AIT, in 2004. In 2003, he was with the Imaging Research Group, Loughborough University, U.K. as an Academic Visitor. He was with Brunel University from 2004 to 2006. He is currently a Senior Research Fellow with the Center for Vision, Speech, and Signal Processing group of the University of Surrey, U.K. His research interests are in video coding, video communication, QoE, and context-aware content adaptation. He has authored over 70 peer reviewed journal and conference papers.

MUHAMMAD ALI IMRAN (M'03–SM'12) received the M.Sc. (Hons.) and the Ph.D. degrees from Imperial College London, London, U.K., in 2002 and 2007, respectively. He was with University of Surrey, U.K. from 2007 to 2016. He has led a number of multimillion international research projects encompassing the areas of energy efficiency, fundamental performance limits, sensor networks, and self-organizing cellular networks. He is currently a Professor of Communication Systems with the University of Glasgow, U.K., and the Vice Dean of the School of Engineering, Glasgow College UESTC. He is also a Visiting Professor with the University of Surrey, U.K., and the University of Oklahoma, USA. He has a global collaborative research network spanning both academia and key industrial players in the field of wireless communications. He has supervised more than 20 successful Ph.D. graduates. He holds ten patents and published over 150 peer-reviewed research papers including over 20 IEEE Transaction papers. He was a recipient of the IEEE Comsoc's Fred Ellersick Award 2014 and the FEPS Learning and Teaching Award 2014 and twice nominated for the Tony Jean's Inspirational Teaching Award. He was a shortlisted finalist for the Wharton-QS Stars Awards 2014 for innovative teaching and the VC's Learning and Teaching Award in University of Surrey. He is a Senior Fellow of Higher Education Academy, U.K.

PEI XIAO (SM'11) received the Ph.D. degree from the Chalmers University of Technology, Sweden, in 2004. He was as a Research Fellow with Queen's University Belfast and had held positions with Nokia Networks, Finland. He is a Reader with the University of Surrey and also the Technical Manager of 5G Innovation Center, leading and coordinating research activities in all the work areas in 5GIC. His research interests and expertise span a wide range of areas in communications theory and signal processing for wireless communications.