Joint Source-Channel Decoding of Polar Codes for HEVC based Video Streaming

JINZHI LIN, Shenzhen Institute of Information Technology, China
YUN ZHANG and NA LI, Shenzhen Institute of Advanced Technology, Chinese Academy of Sciences, China
HONGLING JIANG, Beijing Information Science & Technology University, China

Ultra High-Definition (UHD) and Virtual Reality (VR) video streaming over 5G networks are emerging, in which High-Efficiency Video Coding (HEVC) is used as source coding to compress videos more efficiently and polar code is used as channel coding to transmit bitstream reliably over an error-prone channel. In this paper, a novel Joint Source-Channel Decoding (JSCD) of polar codes for HEVC based video streaming is presented to improve the streaming reliability and visual quality. Firstly, a Kernel Density Estimation (KDE) fitting approach is proposed to estimate the positions of error channel decoded bits. Secondly, a modified polar decoder called R-SCFlip is designed to improve the channel decoding accuracy. Finally, to combine the KDE estimator and the R-SCFlip decoder together, the JSCD scheme is implemented in an iterative process. Extensive experimental results reveal that, compared to the conventional methods without JSCD, the error data-frame correction ratios are increased. Averagely, 1.07% and 1.11% Frame Error Ratio (FER) improvements have been achieved for Additive White Gaussian Noise (AWGN) and Rayleigh fading channels respectively. Meanwhile, the qualities of the recovered videos are significantly improved. For the 2D videos, the average Peak Signal-to-Noise Ratio (PSNR) and Structural SIMilarity (SSIM) gains reach 14% and 34% respectively. For the 360° videos, the average improvements in terms of Weighted-to-Spherically-uniform PSNR (WS-PSNR) and Voronoi-based Video Multimethod Assessment Fusion (VI-VMAF) reach 21% and 7% respectively.

CCS Concepts: • **Information systems** → **Multimedia streaming**; • **Mathematics of computing** → **Coding theory**; • **Networks** → *Mobile networks.*

Additional Key Words and Phrases: joint source-channel decoding, HEVC, polar code, video streaming

ACM Reference Format:
Jinzhi Lin, Yun Zhang, Na Li, and Hongling Jiang. 2021. Joint Source-Channel Decoding of Polar Codes for HEVC based Video Streaming. 1, 1 (November 2021), 23 pages. https://doi.org/10.1145/1122445.1122456

This work was supported in part by the National Natural Science Foundation of China under Grant 62172400 and 61902389, in part by the Shenzhen Science and Technology Program under Grant JCYJ20180507183823045 and JCYJ20200109110410133, in part by Guangdong International Science and Technology Cooperative Research Project under Grant 2018A050506063, in part by Membership of Youth Innovation Promotion Association, Chinese Academy of Sciences under Grant 2018392, in part by the Beijing Municipal Education Commission Applied Basic Research Project under Grant KM202011232022.
Authors' addresses: J. Lin, Shenzhen Institute of Information Technology, Shenzhen, China, 518172; email: linjz@sziit.edu.cn; Y. Zhang(corresponding author) and N Li, Shenzhen Institute of Advanced Technology, Chinese Academy ofSciences, Shenzhen, China, 518055; emails: {yun.zhang, na.li1}@siat.ac.cn; H. Jiang, Beijing Information Science & Technology University,Beijing, China, 10092; email: jhl@bistu.edu.cn.

XXXX-XXXX/2021/11-ART $15.00
https://doi.org/10.1145/1122445.1122456

1 INTRODUCTION

Ultra High-Definition (UHD) and Virtual Reality (VR) videos are becoming popular since they are capable of providing more realistic visual experiences. Due to the huge amount of data volume, these videos are usually compressed effectively with highly efficient source coding, such as H.265/High-Efficiency Video Coding (HEVC), which doubles the compression ratio as compared to the H.264/Advanced Video Coding (AVC). Even so, wide bandwidth is required to stream the compressed videos. Thanks to the development of network transmission, the 5th Generation (5G) mobile transmission technologies significantly improve the transmission bandwidth and lower the delay. However, there is still a big gap between the bandwidth provided by the existing communication networks and the bandwidth required by UHD and VR video streaming.

Due to channel noises, signal interference, and multi-path fading, *etc.*, bitstreams transmitted over wireless channels are error-prone. Channel coding or Forward Error Correction (FEC) [50] is applied for error controlling [20]. The polar code is one of the channel coding schemes adopted in the 5G standard [1], which will probably be further utilized in 5G data plane as it is more suitable for long codes and more effective for massive data coding. Conventionally, source and channel en/decoding are optimized separately, which are expounded extensively by Shannon [45]. However, Shannon's separation theorems rely on some assumptions, including infinite code length, no delay feedback, and infinite feedback capacity, which may not be guaranteed in a practical system. For practical applications, separate source-channel en/decoding limits system performance [12]. The redundancies hidden in the source coding are the extra extrinsic information for channel coding, which can be utilized for improving channel decoding accuracy. Methods based on this idea are Joint Source-Channel Decoding (JSCD) approaches.

For the emerging VR broadcasting application, the data rate of a 360° video that allows a full 360° high quality viewing experience requires about 400 Mb/s [53]. The reasonable motion-to-photon delay is shall be below 15-20 milliseconds [42]. Even transmitting over 5G networks, the wide bandwidth and low latency requirements are challenging to VR video streaming. To tackle these challenges, viewport-specific [51] and tiled based [37] streaming schemes were developed. Besides, another challenging issue is to handle the error-prone bitstreams to improve data transmission efficiency. Automatic repeat requests could be employed for re-transmitting error bitstreams. However, they consume extra communication resources. The error concealment/error resilience [24] technologies provided by AVC and HEVC decoders reduce the negative impact from errors. However, they brought an extra computational complexity and the quality gain from recovering was still limited. Many researches show that, with the cost of consuming a few additional computation, a JSCD method can improve data transmission efficiency without extra network bandwidths. Since bandwidths are the bottleneck of a network and are hard to be expanded, the JSCD method suits for a practical video streaming system. In this paper, we propose a novel JSCD scheme of polar codes for HEVC based video streaming, which creatively improves the accuracy of the polar channel decoding by exploiting the HEVC bitstreams syntax. The unique challenges of the proposed scheme include: how to find the redundant source information from the HEVC coding standard and how can they be utilized to improve the polar channel decoding performance. The proposed method is specifically suitable for video playing end-devices with rich computation abilities, which at the same time demands high definition videos recovered from remote streaming servers. To focus on JSCD related processes, video encryption and adaptive video streaming technologies are not considered in this paper. The contributions of this paper are given as follows:

- A Kernel Density Estimation (KDE) fitting approach for estimating positions of error channel decoded bits from HEVC bitstreams is proposed, based on the analysis of HEVC syntax error types and statistics of the error positions.

- An algorithm called R-SCFlip for decoding data-frames [1] with polar codes is designed, which improves the decoding accuracy compared to the original successive cancellation flip decoding algorithm.
- An iterative JSCD scheme is proposed by combining the KDE error bit range estimator and the R-SCFlip decoder, which improves reliability and visual quality of the video streaming.

The rest of this paper is organized as follows. Firstly, Section 2 reviews the related works. Section 3 briefly introduces the fundamentals of polar codes and the corresponding successive cancellation decoder. Then, the proposed JSCD scheme is presented in Section 4. Section 5 presents the experimental results and analysis. Finally, Section 6 draws the conclusion.

2 RELATED WORKS

2.1 General JSCD, JSCC and Cross-layer Schemes

There are a number of JSCD schemes for sensor data and controlling signal transmitting. Dumitrescu *et al.* [14] dealt with Markov sequence sources. JSCD approaches were proposed for accelerating both the Maximum A Posteriori (MAP) sequence decoding and the soft output Max-Log-MAP decoding when the Markov sources satisfied Monge property. Two correlated sensor data encoded by systematic LDPCs independently were considered in [28]. A JSCD decoder composed of two LDPC decoders was proposed, where the encoded bits at the output of each LDPC decoder were used as the *a priori* information at the other decoder. Abdessalem *et al.* [2] presented similar ideas by considering relay cooperative communications. Methods of joint channel decoding and state estimation for cyber-physical systems have been studied in [19].

Opposite to JSCD, Joint Source-Channel Coding (JSCC) combined source encoding with channel FEC codes. JSCC schemes are usually demanded for designing the corresponding JSCD approaches. Coupling with multimedia broadcasting, the authors have studied multilevel coded modulation for providing Unequal Error Protection (UEP) JSCC approach in [48]. Liu *et al.* [35] formulated error-resilient VR video transmission into JSCC optimization problem and solved it by coming up a heuristic algorithm. In [5], a hybrid JSCC scheme with binary and non-binary turbo codes was introduced, which utilized techniques including JSCD, regression-based extrinsic information scaling, and prioritized 16-quadrature amplitude modulation. In fact, Hybrid Digital Analog (HDA) coding can be viewed as another form of JSCC. Recently, some works using HDA coding for UHD and 3-D videos transmission are emerging [33, 36]. Deep learning based approaches have been investigated to improve the JSCC and JSCD [7, 31, 34], which brought interesting inspirations and performance improvements.

In a cross-layer perspective, the basic principles and design highlights of JSCD considering coupling multiple protocol layers for video communicated in a wireless network were discussed in [13]. Qiwang *et al.* reviewed integrated physical-layer and cross-layer communication coding systems for optimization of component elements to achieve green communications [9]. A cross-layer optimization of caching and delivery control for minimizing the overall video delivery time in two-hop relaying networks was investigated in [49]. Cross-layer optimization framework for maximizing the total utility of 360° videos delivering in the multi-cast systems was proposed in [40]. Cross-layer design techniques in wireless multimedia sensor networks for energy conservation were studied in [29]. Zhu *et al.* proposed a joint layered approach to achieve reliable and secure JPEG-2000 image streaming over mobile networks [52]. More cross-layer design methods dedicated in video streaming were presented in [10, 17].

[1] To distinguish the concepts of a "frame" in a video and a "frame" in the communication physical layer, the word "data-frame" is used in the whole paper referring to the physical layer frame. Otherwise, "frame" refers to a video frame.

2.2 Video Streaming Dedicated JSCD Schemes

Different kinds of H.264/AVC based JSCD schemes have been proposed in [22, 30, 32, 38, 47]. In general, the last step of the video source coding is the arithmetic coding, such as variable length coding and Context-Adaptive Binary Arithmetic Coding (CABAC) for H.264. It is straightforward to practice JSCD by tight coupling arithmetic coding with channel coding. Based on MAP estimation, Wang *et al.* [47] proposed a JSCD method for variable length coding and convolutional encoded 1-D Markov source, and applied it to decode motion vectors of H.264 coded video streams. In [32], an iterative JSCD scheme for videos encoded by H.264 with CABAC entropy coding and a rate-1/2 Recursive Systematic Convolutional (RSC) code was proposed. In this scheme, slice candidates with different likelihoods were generated and checked for source semantic validation. The bitstreams, corresponding to invalid slice candidates, were modified and fed back to the soft output viterbi algorithm decoder to do channel decoding recurrently. This scheme was extended in [38] by introducing a virtual checking method to accelerate the semantic verification process. Hanzo *et al.* [22] presented various Short Block Codes (SBC) based iterative JSCD system design for near-capacity H.264 source coded videos, and proposed a redundant source mapping scheme that can improve the convergence behavior of SBCs. In [30], a sequential MAP JSCD scheme was proposed. The scheme utilized forbidden symbols in arithmetic coding, semantics and syntax validation checking and *a priori* probability estimation for syntax element sequences to implement the core idea of JSCD. The above mentioned works focus on JSCD approaches dedicated to H.264. As HEVC is designed to focus on increasing video resolution and parallel processing, the semantics and syntax of HEVC are somehow different from its predecessor H.264. The existing methods of detecting error bits in bitstreams for H.264 cannot be applied to HEVC directly.

HEVC related JSCD schemes can be found in [25, 41]. Perera *et al.* [41] designed a cross-layer turbo decoder to make use of HEVC source redundancy in the form of exploiting slice header semantics and field patterns. Specifically, a Slice-header-field Parsing and Correction (SPC) module for checking the HEVC syntax was required. To implement SPC, algorithms of access unit boundary identification and Network Abstract Layer (NAL) unit header fields analyzing and amending were proposed in the paper. As a boundary identification algorithm for access units was used, the reported error bit positions were always accurate. The disadvantage of this method is that it can only be applied to a few numbers of semantic and syntax error types. In [25], not constrained to any specific video coding standards and modeling the correlation inherent in compressed video signals as a first-order Markov process, Huo *et al.* proposed a spatio-temporal iterative JSCD system.

2.3 Polar Code Related JSCD Schemes

Most existing JSCD schemes were developed based on the traditional channel codes, such as turbo codes and Low-Density Parity-Check (LDPC) codes. However, only a few JSCD related studies were on the latest polar codes adopted by the 5G communication standard. A polar code is constructed by a channel polarization transform process which is completely different from the LDPC and turbo codes. The existing JSCD schemes with LDPC or turbo codes are not suitable for polar codes. Jin *et al.* [27] proposed a distributed JSCD scheme for tackling general correlated sources encoded by systematic polar codes independently, and proposed a joint source-channel polarization scheme by using a quasi-uniform systematic polar code [26]. A JSCD method for language-based sources with polar encoding was exploited in [46]. Source redundancy was utilized by judging the validity of the decoded words in the decoded sequence with the help of a dictionary. The scenario that HEVC and polar codes are involved in source and channel coding respectively is not tackled in the above mentioned works. As for JSCC related polar codes, Jin *et al.* [26] considered joint source and channel polarization and Hadi *et al.* [21] utilized the channel polarisation property to achieve UEP.

In summary, JSCD schemes for video streaming are different from conventional data streaming since videos have large data volume and are compressed in lossy. It is required to develop JSCD by considering the properties of both HEVC bitstream and polar code.

3 PRELIMINARIES OF POLAR CODES

Let a vector $[x_1, \cdots, x_N]$ be x_1^N. Consider to transmit K bit information over the communication channel, a polar code of length $N = 2^n, N > K$ with rate $R = K/N$ separates the N synthetic polarized channels into K reliable and $N - K$ unreliable ones, and encode the information bits and frozen bits on them respectively. Denote $\mathcal{I}$ as the set containing the indices of the K reliable synthetic channels. The encoding process can be described as

$$\boldsymbol{X} = \boldsymbol{U} \cdot G^{\otimes n} \tag{1}$$

where $\boldsymbol{U} = u_1^N = [u_1, \cdots, u_N]$ of length N is the input data vector, containing K information bits at position $i \in \mathcal{I}$ and $N - K$ frozen bits that are set to zero. $\boldsymbol{X} = x_1^N = [x_1, \cdots, x_N]$ is the encoded vector. $G^{\otimes n}$ is the generator matrix which is the $n-th$ Kronecker product of the polarization matrix $G = \begin{bmatrix} 1 & 0 \\ 1 & 1 \end{bmatrix}$.

The original standard polar code decoding algorithm is Successive Cancellation (SC) decoder proposed by Arikan [4]. Denote the received signal from channel as $\boldsymbol{Y} = y_1^N = [y_1, \cdots, y_N]$ which is the noisy version of $\boldsymbol{X}$. SC decoder tries to recover u_1^N via successively decoding u_i in ascending order of index i. For u_i, its LLR is computed by

$$L(u_i) = \log\left(\frac{P(u_i = 0|\boldsymbol{Y}, u_1^{i-1})}{P(u_i = 1|\boldsymbol{Y}, u_1^{i-1})}\right) \tag{2}$$

According to the sign of its LLR value and whether it is a frozen bit, u_i is decoded as $\hat{u}_i$

$$\hat{u}_i = \begin{cases} 1 & \text{if } i \in \mathcal{I} \text{ and } L(u_i) < 0 \\ 0 & \text{if } i \notin \mathcal{I} \text{ or } L(u_i) \geqslant 0 \end{cases} \tag{3}$$

4 PROPOSED JOINT SOURCE-CHANNEL DECODING SCHEME

4.1 System Model

Fig. 1 depicts a video streaming system consists of source video compression, channel encoding, transmission, channel decoding and video decoding. The input video sequence is firstly put into a HEVC encoder, a sequence of syntax elements $v_1^{\ell_v} = [v_1, \cdots, v_{\ell_v}]$ of length ℓ_v is produced after coding, where v_i is a syntax element which is placed into a NAL unit. Then, these syntax elements are mapped into a binary sequence $s_1^{\ell_s} = [s_1, \cdots, s_{\ell_s}]$ of length ℓ_s, where s_i is either 0 or 1. Next, the entropy encoder CABAC compresses $s_1^{\ell_s}$ to a bit sequence $u_1^{\ell_u} = [u_1, \cdots, u_{\ell_u}]$ of length ℓ_u, where u_i is also either 0 or 1. Before transmission, bit sequence $u_1^{\ell_u}$ is encoded with a polar code channel encoder, and the obtained encoded vector $x_1^{\ell_x} = [x_1, \cdots, x_{\ell_x}]$ of length ℓ_x is modulated and sent through a wireless noisy channel. At the receiver side, the received signal vector $y_1^{\ell_y} = [y_1, \cdots, y_{\ell_y}]$ is the result of $x_1^{\ell_x}$ corrupted by noise $\boldsymbol{n}$. The goal of the whole decoding process at the receiver is to recover $u_1^{\ell_u}$, $s_1^{\ell_s}$ and $v_1^{\ell_v}$ as best as possible, the estimated value of them are denoted as $\hat{u}_1^{\ell_u}$, $\hat{s}_1^{\ell_s}$ and $\hat{v}_1^{\ell_v}$, respectively.

The goal of the polar decoder is to estimate $u_1^{\ell_u}$ by maximizing *a posteriori* probability as

$$\hat{u}_1^{\ell_u} = \underset{\tilde{u}_1^{\ell_u} \in \mathcal{B}_{\ell_u}}{\arg\max}\, P(\tilde{u}_1^{\ell_u}|y_1^{\ell_y}) \tag{4}$$

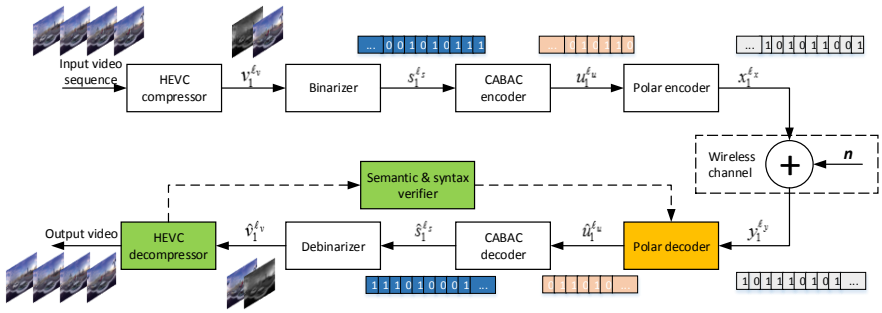


Fig. 1. Wireless video streaming system model.

where $\mathcal{B}_{\ell_u}$ is the set of all the bit sequences with a length of ℓ_u. Using the Bayes' rule, it can be written as

$$\hat{u}_1^{\ell_u} = \arg\max_{\tilde{u}_1^{\ell_u} \in \mathcal{B}_{\ell_u}} \frac{P(y_1^{\ell_y}|\tilde{u}_1^{\ell_u})P(\tilde{u}_1^{\ell_u})}{P(y_1^{\ell_y})} \tag{5}$$

There are three terms in the right side of the equation: channel transition probability $P(y_1^{\ell_y}|\tilde{u}_1^{\ell_u})$ depending on the physical characteristics of the communication channel, modulation method and polar encoding; *a priori* probability $P(\tilde{u}_1^{\ell_u})$ of bit sequence $u_1^{\ell_u}$ determined by the distribution of binary sequence $s_1^{\ell_s}$; and the denominator $P(y_1^{\ell_y})$ which is constant for all realizations of $\tilde{u}_1^{\ell_u}$ and thus is insignificant in this maximization.

For source-channel separation decoding methodologies, $P(\tilde{u}_1^{\ell_u})$ is unknown to the channel decoder, thus $\tilde{u}_1^{\ell_u}$ are assumed to be Independent and Identically Distributed (IID) and Bernoulli(0.5) distribution is usually adopted [18]. However, for HEVC compressed video source, $\tilde{u}_1^{\ell_u}$ are corresponding to $s_1^{\ell_s}$ which are constrained by HEVC syntax elements $v_1^{\ell_v}$. Not all $\tilde{u}_1^{\ell_u} \in \mathcal{B}_{\ell_u}$ are necessary to be valid bit sequences satisfying HEVC semantic and syntax constraints. Information hidden in $P(\tilde{u}_1^{\ell_u})$ can be utilized to improve the MAP decoding performance. To obtain a specific probability distribution of $\tilde{u}_1^{\ell_u}$ by observing $v_1^{\ell_v}(\tilde{u}_1^{\ell_u})$ is unpractical. Therefore, implementing JSCD via MAP decoding is not feasible. In this paper, to make use of the extrinsic information from the syntax elements, a model to estimate error bit location range based on HEVC semantic and syntax verification (shown as the green blocks in Fig. 1) and an improved SCFlip polar decoder to utilize the estimated ranges (shown as the yellow block in Fig. 1) are designed.

4.2 Error Bit Location Range Estimation

In HEVC, encoded bitstreams are organized in a bunch of NAL units separated (or synchronized) by start codes. Each NAL unit [44] consists of a header and the associated payload data called Raw Byte Sequence Payload (RBSP), where some logical syntax elements are presented together. The six bits NALType field in the NAL header specifies the role of the unit and determines the format of the unit's RBSP. The syntax elements constructing all types of RBSPs are well defined in the HEVC standard, along with some semantic constraints if necessary.

In a video streaming system, NAL units are further channel encoded into data-frames and transmitted through noisy communication channels. At the decoder side, the HEVC decoder may confront that some syntax elements are not compliant with the standard due to error bits. In this case, syntax errors will be reported. Moreover, the decoded values of some syntax elements may

violate the semantics in standard, i.e., out of valid range, illegal values, undefined meaning, etc., then semantic errors may be reported as well.

Video decoding errors could be propagated and cumulated due to intra/inter predictive coding. Information in the headers of NAL units is relatively important. Error bits inside them could cause their following RBSP cannot be decoded properly, or even cause decoding corruption for some referring units. As most percentages of a bitstream are data slices encoded by CABAC [44]. They are extremely sensitive to bit errors due to context correlation and changing probabilities, which cause desynchronization and value deviation problems. When the decoder reports semantic or syntax errors in decoding a bitstream at some specific positions, they are not meant to be the exact locations where the actual errors bits occur. Instead, the error bits may locate nearby. To overcome this problem, a statistical approach for error bit location range estimation is proposed.

In the reference software HEVC test Model (HM) [39], there are many assertion statements associated with all kinds of HEVC semantic and syntax validation. Generally, an assertion failure is considered as a report of a semantic and syntax error. Table 1 lists the most common semantic and syntax errors that the HM software may report in decoding bitstreams. For example, because HEVC standard defines the first bit in a NAL unit's header to be zero, if the HM software encounters a non-zero value in that, it would report a *forbidden_zero_bit* ! = *0* semantic error.

Table 1. The most common semantic and syntax errors reported by the HM software.

Error type	In head	In RBSP	Sem./Syn.
forbidden_zero_bit ! = 0	✓		Semantic
PPS == null	✓		Semantic
invalid sliceQP	✓		Semantic
numAllocatedSlice ! = SliceIdx	✓		Syntax
invalid nalUnitType	✓		Semantic
bits not byte aligned	✓	✓	Syntax
fifo_idx ⩾ fifo.size	✓	✓	Syntax
end_of_slice_segment_flag ! = 1		✓	Semantic
trailingNullByte ! = 0		✓	Semantic
trailing_zero_8bits ! = 0		✓	Semantic
rbsp_stop_one_bit ! = 1		✓	Semantic

We conduct the following experiment. Flip a random bit of a bitstream deliberately, then put it into the HM software for HEVC decoding. An assertion failure may be reported when the flipped bit causes a semantic or syntax error. The position of the current bit being decoded were recorded. Denote the recorded bit position and the actual flipped bit position as p' and p, respectively. Call the difference between p' and p as Causality Position Deviation (CPD), denoted as:

$$\Delta_p = p' - p \tag{6}$$

For a ground truth error bit location estimator, CPDs are always in the predicted ranges. This can be done by predicting a larger range. However, for a practical estimator, the predicted ranges should be as narrow as possible. Small slice sizes in HEVC encoding are associated with narrow predicted ranges. However, it is inefficient to use a very small slice size, since it may include more slice head overhead. The value of 100 bytes is used as the slice size, which is a trade-off achieved from statistical experiments. We collected the statistic of CPDs of all kinds of assertion failures. Fig. 2 shows the empirical Cumulative Distribution Function (CDF) of CPDs for syntax error "fifo_idx ⩾ fifo.size" when bitstreams are encoded with slice size set to 100 bytes. According to which specific parameter is currently being decoded, the errors are further divided into several categories, such as

decodeSplitFlag, decodeCoeff, decodePredInfo, and others. In general, most of the CPDs are within range $[-600, 2000]$ and the ranges vary with categories.

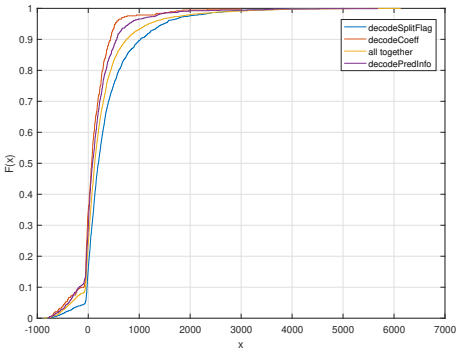


Fig. 2. Empirical CDF of CPDs for syntax error "fifo_idx ⩾ fifo.size".

Actually, the distributions of CPDs can be accurately fitted by the KDE method [43]:

$$\hat{f}_h(x) = \frac{1}{nh} \sum_{i=1}^{n} K\left(\frac{x - x_i}{h}\right) \tag{7}$$

where $x_i, i = 1 \cdots n$ are the sampled data, n is the sampled data size, $K(\cdot)$ is the kernel smoothing function and Epanechnikov [15] function $K(u) = \frac{3}{4}\left(1 - u^2\right)$is used in this paper, h is the bandwidth. Utilizing experimental data, $\hat{f}_h(x)$ for CPDs of different semantic and syntax errors can be obtained. The estimator predicts the error bit location ranges as

$$\begin{aligned} p_{LB} &= p' + \hat{F}_h^{-1}(\sigma_{LB}), \\ p_{UB} &= p' + \hat{F}_h^{-1}(\sigma_{UB}), \\ \hat{F}_h(x) &= \tfrac{1}{n} \sum_{i=1}^{n} G\left(\tfrac{x - x_i}{h}\right), \\ G(x) &= \textstyle\int_{-\infty}^{x} K(t)dt \end{aligned} \tag{8}$$

where $\hat{F}_h(x)$ is the CDF of $\hat{f}_h(x)$, σ_{LB} and σ_{UB} are the parameters to be configured, which determine the possibility that the error bits located in the predicted range. To obtain a 0.95 possibility, 0.025 and 0.975 are used in this paper, respectively. The predicted range is denoted as $[p_{LB}, p_{UB}]$ by indicating the lower and upper bounds of error bit position.

To implement the KDE model based error bit location estimator, a large number of CPD samples are collected through experiments. By applying KDE fitting to these samples, the bandwidths and distributions for different semantic and syntax errors are obtained. Next, the estimated ranges are calculated according to equation (8). Table 2 gives the adopted range predictions and the corresponding accuracy for different semantic and syntax errors. The predicted ranges of items with h indicated by '-' are not configured by KDE fitting, as there are not enough samples during the experiments due to their rare occurring, and their $[p_{LB}, p_{UB}]$ values are simply set by the minimum and maximum of their few corresponding samples respectively. Note that the error bit location range estimator is previously established before running the JSCD process. There is no need to run KDE fitting for each specific video during conducting JSCD.

Table 2. Configuration of KDE parameters for common syntax errors ($\sigma_{LB} = 0.025, \sigma_{UB} = 0.975$).

Error type	h	$[p_{LB}, p_{UB}]$	Acc. (%)
forbidden_zero_bit ! = 0	-	$[p', p']$	100
PPS == null	8.32	$[p' - 32, p' + 11]$	100
invalid sliceQP	-	$[p' - 34, p' - 2]$	100
numAllocatedSlice ! = SliceIdx	1.11	$[p' - 39, p' - 25]$	98
invalid nalUnitType	-	$[p' - 27, p' - 27]$	100
bits not byte aligned	-	$[p' - 56, p' - 2]$	100
fifo_idx ⩾ fifo.size (decodeSplitFlag)	42.84	$[p' - 606, p' + 1539]$	95
fifo_idx ⩾ fifo.size (decodeSkipFlag)	36.78	$[p' - 652, p' + 555]$	94
fifo_idx ⩾ fifo.size (decodePredMode)	53.13	$[p' - 504, p' + 598]$	93
fifo_idx ⩾ fifo.size (decodePartSize)	39.79	$[p' - 624, p' + 848]$	88
fifo_idx ⩾ fifo.size (decodeCoeff)	37.32	$[p' - 366, p' + 2012]$	89
fifo_idx ⩾ fifo.size (decodeMergeIndex)	48.09	$[p' - 715, p' + 514]$	94
fifo_idx ⩾ fifo.size (decodePredInfo)	32.15	$[p' - 549, p' + 1284]$	89
fifo_idx ⩾ fifo.size (others)	2.93	$[p' - 75, p' + 8]$	100
end_of_slice_segment_flag ! = 1	16.82	$[p' - 423, p' + 525]$	92
trailingNullByte ! = 0	1.51	$[p' - 781, p' + 44]$	94
trailing_zero_8bits ! = 0	104.34	$[p' - 1067, p' - 59]$	94
rbsp_stop_one_bit ! = 1	20.07	$[p' - 662, p' + 405]$	91

4.3 Range Specified Successive Cancellation Flip Decoding

In the standard SC polar code decoder, due to the sequential property, intermediate bit decoding errors can propagate through the subsequent bits decoding. It was observed in [3] that when one or more incorrect bit estimations happen, it can cause more incorrect estimations in the subsequent decoding. The Successive Cancellation Flip (SCFlip) decoding [16] was proposed to find and flip the first falsely decoded bit, hoping that its follow-up error bits caused by this incorrect estimation can be corrected. In the original SCFlip decoder, bits to be flipped are determined by their LLR values, which are the smallest ones among all the un-frozen bits. In fact, this operation is not guaranteed to be true. The bit with the smallest LLR value is not necessary the first error bit to be flipped for the next SC decoding trial. We can imagine that, when the SCFlip decoder searches bits to be flipped only within a previously known range, where the actual first error bit may locate in, the decoding performance can be improved. We call this range specified SCFlip decoding as R-SCFlip.

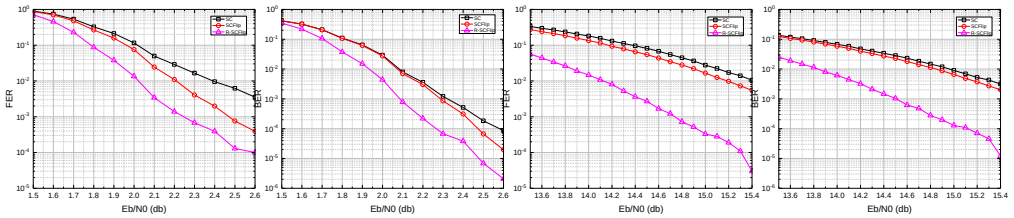
Given a polar code with length N for sending K information bits, in which r CRC bits for checking the validity of the decoded codewords are included. The sending codeword and received signal are denoted as u_1^N and y_1^N, respectively. Algorithm 1 gives the pseudo-code of R-SCFlip, in that, function SC $\left(y_1^N, \mathcal{A}, k\right)$ stands for the SC algorithm based on the received signal y_1^N and the set of non-frozen bits $\mathcal{A}$, with bit $\hat{u}_k$ flipped. Similar to the SCFlip decoder, R-SCFlip starts by performing a standard SC process to gain the first estimation of $\hat{u}_1^N$, as well as their corresponding LLR values $L\left(y_1^N, \hat{u}_1^{i-1} | u_i\right)$. If $\hat{u}_1^N$ passes the CRC checking, the decoding finishes. Otherwise, R-SCFlip would attempt to find out the T most unreliable bits (denote $\mathcal{U}$ as the set of the indices of them) within the given range $\mathcal{R}$ according to $L\left(y_1^N, \hat{u}_1^{i-1} | u_i\right)$. Then for every bit $\hat{u}_k, k \in \mathcal{U}$, R-SCFlip is given a chance to do the SC decoding, in that $\hat{u}_k$ is flipped with respect to its decoded result in the standard SC algorithm, the flipped value of $\hat{u}_k$ is fed forward to take part in calculating the LLRs of the following decoding bits, thus affect the whole decoding result $\hat{u}_1^N$. Again, if the newly decoded $\hat{u}_1^N$ passes the CRC checking, the decoding is completed. Otherwise, R-SCFlip continues the process until all the T chances have been tried out.

The difference between SCFlip and R-SCFlip lies in that the ranges to find the first error bit for flipping are different. R-SCFlip identifies the most unreliable bits by searching the smallest $L\left(y_1^N, \hat{u}_1^{i-1} | u_i\right)$ values in the given range $\mathcal{R}$, while SCFlip searches the whole range of the current

Algorithm 1: The proposed R-SCFlip decoding.

Input : $y_1^N, \mathcal{A}, T, \mathcal{R}$
Output: $\hat{u}_1^N$
$(\hat{u}_1^N, L(y_1^N, \hat{u}_1^{i-1}|u_i)) \leftarrow \text{SC}(y_1^N, \mathcal{A}, 0)$;
if $T > 1$ *and* $CRC(\hat{u}_1^N) ==$ *failure* **then**
  $\mathcal{U} \leftarrow i \in (\mathcal{A} \cap \mathcal{R})$ of T smallest $\left|L(y_1^N, \hat{u}_1^{i-1}|u_i)\right|$;
  $j = 1$;
  while $j \leq T$ **do**
    $k \leftarrow \mathcal{U}(j)$;
    $\hat{u}_1^N \leftarrow \text{SC}(y_1^N, \mathcal{A}, k)$;
    if $CRC(\hat{u}_1^N) ==$ *success* **then**
      break;
    $j = j + 1$;
  end

decoding data-frame. As can be expected, if the actual first error bit does locate in the given range, R-SCFlip has a higher possibility of flipping the exact error bits than SCFlip and tends to be more likely to render the correct decoding result. Thus, R-SCFlip is expected to be able to correct more data-frames than the SCFlip. Fig. 3 shows the simulation performances of SC, SCFlip, and R-SCFlip decoders. The polar code related parameters configured in the simulations are the same as in the experiments in section 5 given in Table 3. Two of the most common channel models, Additive White Gaussian Noise (AWGN) and Rayleigh fading channels, are considered. The simulation Signal-to-Noise Ratio (SNR) ranges for the two channels are set to $1.5 \sim 2.6$ and $13.5 \sim 15.4$ db (measured in $Eb/N0$), respectively. The fading channel is more realistic than the AWGN channel, experiments in these two channel models help to prove that the proposal can work both in ideal and practical channels. As can be seen from Fig. 3, the R-SCFlip decoder has the smallest Frame Error Ratio (FER, known as the percentage of error data-frames) and Bit Error Ratio (BER). R-SCFlip can significantly decrease the FER and BER for the SCFlip decoder in AWGN and fading channels.



(a) FER, AWGN channel (b) BER, AWGN channel (c) FER, fading channel (d) BER, fading channel

Fig. 3. FER and BER of SC, SCFlip and R-SCFlip decoders in AWGN and Rayleigh fading channels.

4.4 Joint Range Estimation and R-SCFlip Decoding

By combining the KDE error bit location estimation and R-SCFlip polar decoding together, an iterative JSCD scheme is proposed. As depicted in Fig. 4, initially, the channel encoded data-frames are gathered by demodulating wireless channel signals. They are channel decoded by performing SCFlip decoding, which is actually implemented by R-SCFlip decoding with the range R set to the whole data-frame $[0, MAX]$. By gathering all the R-SCFlip decoded data-frames together, the raw video source encoded bitstream is obtained. As the bitstream is composed of NAL units, the

video decoder separates the bitstream into a list of NAL units L_{NALs} by performing NAL boundary identification.

The following process is to do JSCD decoding in an iterative form for the NAL units in L_{NALs} one by one. The current NAL unit to be decoded is denoted as NAL_{cur} and is assigned to the elements of L_{NALs} iteratively by invoking Next(L_{NALs}). If it is not null (meaning that there still remains units to be processed), it is passed to perform JSCD. Otherwise, the iterative JSCD process finishes. Firstly, NAL_{cur} is fed into the video decoder to do HEVC decoding. If there are no assertion failures reported by the decoder, then the decoded source bits for NAL_{cur} are assumed to be correct and move on to the next NAL unit. Otherwise, according to the reported assertion message and the position of the current decoding bit in NAL_{cur}, the error type and error bit position p' can be deduced. This information is saved and compared with the result of the last JSCD decoding process. If they are the same, it indicates that the last R-SCFlip channel decoding has failed to correct the error bits, thus the conventional video decoding error concealment is performed and then moves on to the next NAL unit to continue the JSCD process. If they are not the same, it indicates that a new error situation caused by new encountered error bits happens, which means that the information has not yet been utilized by R-SCFlip decoding in doing error bits correction. Therefore, the error bit location range estimation and R-SCFlip decoding should be combined. The data-frame F corresponding to NAL_{cur} is firstly determined. The location range of possible error bits in data-frame F causing the assertion of failure is estimated as $R = [p_{LB}, p_{UB}]$ through the KDE fitting model according to the semantic and syntax error type, as depicted in the previous subsection. With the given estimated range R, data-frame F is fed into R-SCFlip decoder to do channel decoding again. The newly generated NAL_{cur} after R-SCFlip decoding is put back to video decoder to do HEVC decoding again, and a new JSCD process repeats. When all of the NAL units in L_{NALs} pass HEVC decoding without assertion of failure or have been conducted error concealment, the JSCD process finishes.

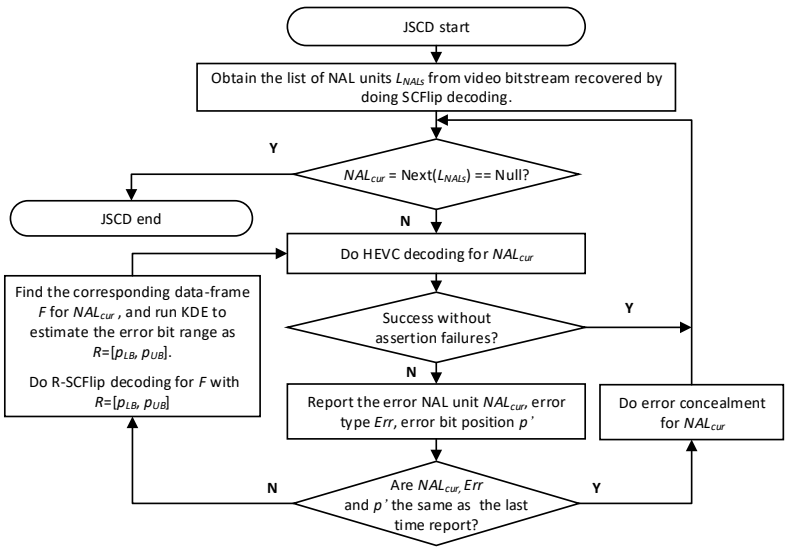


Fig. 4. Flow chart of the proposed iterative JSCD scheme.

In the proposed iterative JSCD, the assertions of failures reported by the HEVC decoding are the outcomes of source bitstream's violation of HEVC semantic and syntax restriction and are essentially the result of information redundancy hidden in HEVC standard. For R-SCFlip channel decoder, they are viewed as external information, which is utilized to help improve channel decoding accuracy. Specifically, in the proposed scheme, it is implemented in an iterative form of estimating position ranges of error bits and doing R-SCFlip polar channel decoding back and forth. The performance of the proposed iterative JSCD mainly relies on the accuracy of the error bit location range estimation and the validation of the R-SCFlip channel decoder.

5 EXPERIMENTAL RESULTS AND ANALYSIS

In this section, extensive experiments are designed to validate the proposed JSCD scheme for transmitting HEVC encoded video bitstreams in the wireless channels with polar encoding. The experimental settings are firstly presented. Then the channel decoding accuracy performance is validated. Next, the video quality improvements are evaluated in terms of different metrics and visual results. Moreover, the computational complexity of the proposal is analyzed and the comparison with a related scheme is given as well.

5.1 Experimental Settings

To simulate polar code en/decoding, the open-source software AFF3CT[8] tool was used. To en/decode VR videos in HEVC, the HM software [6] integrated with 360lib [23] was used. Four 360° video sequences (*AerialCity*, *DrivingInCity*, *DrivingInCountry*, *PoleVault*) representing VR multimedia and two 2D regular videos (*BasketBallDrive*, *RaceHorses*) were chosen for evaluation. They were encoded in HEVC with four encoding Quantization Parameters (QP) 37, 32, 27, 22. Note that, in the following tables, the names of the testing videos are shortened by the abbreviations A.C., D.I.C., D.I.Cnt., P.V., B.D., and R.H. respectively. For wireless communications, the AWGN and Rayleigh fading channels with different Signal-to-Noise Ratio (SNR) levels were considered.

Table 3 summarizes the experimental configurations. For the two 2D regular videos, they were divided into 5 and 3 segments respectively (each segment consists of 100 frames), and all the segments were tested in the experiments. The following experimental results for these two 2D videos are the average values calculated from all of their corresponding segments. It is found that the number of testing video frames has no significant influence on the JSCD schemes' performances[47]. Therefore, for the 360° videos, the first 50 video frames are selected for QP 37 and 32, and the first 25 and 10 video frames are selected for QP 27 and 22, respectively. Detailed information of the testing videos is summarized in Table 4, which gives the number of video frames, bitstream sizes and numbers of NALs (The 2D videos are given in segments). All the experiments with the same parameter configurations are run for 100 and 10 times for the 360° videos and the 2D videos, thus all the presented data are the average values calculated from these individual simulations.

5.2 Analysis on Channel Decoding Accuracy

With the help of the proposed JSCD, the R-SCFlip channel decoder tries to decode data-frames from the physical layer raw video bitstream by flipping error bits in the range predicted by the KDE estimator. Consequently, it gives more chances to recover error decoded data-frames, leading to reduce FER. Suppose N_{frm} is the total number of data-frames, and N_E is the number of error frames without involving JSCD. Among these N_E error frames, some of them are corrected after performing JSCD. Denote the number of corrected frames as $N_E^{'}$. Then, define $\Delta_{FER} = N_E^{'}/N_{frm}$ and $\Delta_{EC} = N_E^{'}/N_E$ as the FER improvement and error data-frame correction ratio, respectively, which indicate the performances of the JSCD method in improving channel decoding accuracy.

Table 3. Summary of the experimental settings.

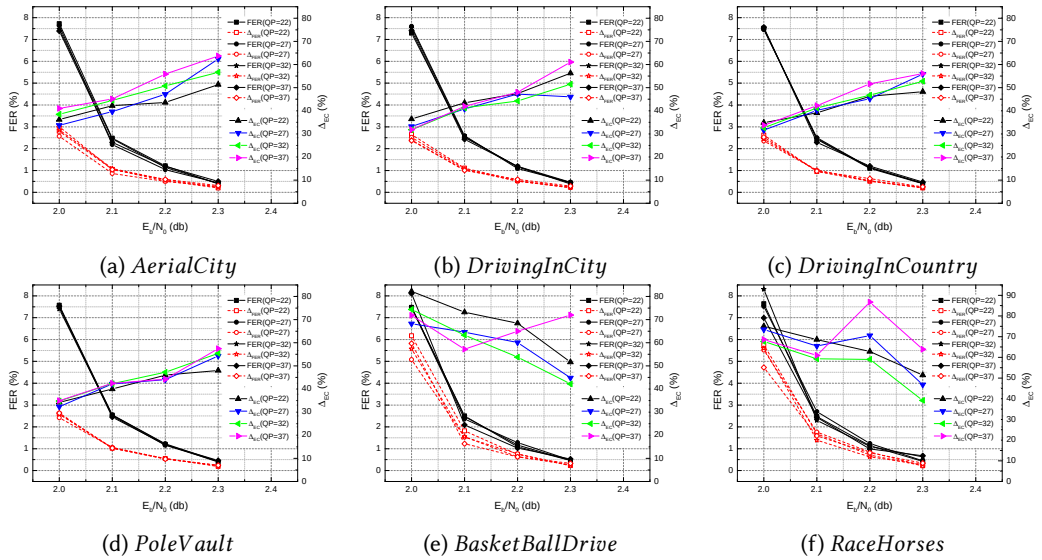
Source coding		Channel coding	
Parameter	Value	Parameter	Value
Video sequence	*AerialCity*, *DrivingInCity*, *DrivingInCountry*, *PoleVault*, *BasketBallDrive*, *RaceHorses*	Polar codeword size (bit)	16384
		Data-frame size (bit)	10240
		Rate of polar code	0.5
Resolution	3840×1920, 1920×1080, 832×480	CRC size (bit)	8
Frames per second	30	Number of SCFlip trials	4
HEVC encoding QP	37, 32, 27, 22	Bandwidth (MHz)	20
GOP size	4	Modulation type	BPSK
Intra period	32	Channel model	AWGN, Rayleigh fading
Max. bytes per slice	100	SNR $\frac{E_b}{N_0}$ (dB)	2.0, 2.1, 2.2, 2.3 (AWGN) 14.6, 14.8, 15.0, 15.2 (fading)

Table 4. Detail summary of testing video sequences.

Seq.	Number of video frames				Bitstream size (bytes)				Nubmer of NALs			
	QP=22	QP=27	QP=32	QP=37	QP=22	QP=27	QP=32	QP=37	QP=22	QP=27	QP=32	QP=37
A.C.	10	25	50	50	1,437,444	683,439	545,609	283,424	8,262	5,596	4,515	2,656
D.I.C.	10	25	50	50	1,200,762	958,512	866,025	445,354	6,839	7,412	7,754	4,449
D.I.Cnt.	10	25	50	50	2,099,940	1,614,957	1,337,263	564,394	9,148	10,973	10,878	5,493
P.V.	10	25	50	50	2,668,564	1,821,949	1,395,539	644,376	11,708	11,747	10,613	5,648
B.D.1	100	100	100	100	5,924,782	2,152,390	1,086,692	589,493	30,131	17,435	10,523	6,227
B.D.2	100	100	100	100	6,930,240	2,412,996	1,186,813	637,157	32,471	18,513	11,087	6,665
B.D.3	100	100	100	100	5,842,612	2,119,768	1,059,965	571,621	28,378	16,422	9,971	6,003
B.D.4	100	100	100	100	5,763,228	2,062,464	1,035,605	555,595	28,441	16,258	9,771	5,786
B.D.5	100	100	100	100	7,025,233	2,392,721	1,181,242	632,550	30,837	17,889	10,840	6,585
R.H.1	100	100	100	100	3,169,178	1,319,091	633,682	304,993	9,840	7,769	4,833	2,966
R.H.2	100	100	100	100	3,122,401	1,223,352	560,953	259,744	9,199	6,801	4,164	2,559
R.H.3	100	100	100	100	1,858,032	848,466	438,942	235,197	8,827	6,361	3,958	2,480

Fig. 5 and 6 give the FER, Δ_{FER} and Δ_{EC} results for the AWGN and Rayleigh fading channels respectively. The black solid lines represent the FERs for different QPs. The red dash lines represent FER improvements Δ_{FER}s and the colorful lines with triangles represent the the error data-frame correction ratios Δ_{EC}s. As can be seen, for all the testing videos with different QPs, Δ_{FER}s are between 0.18% and 3.01% for AWGN channel and between 0.26% and 2.63% for the fading channel. Δ_{EC}s are between 25.08% and 73.41% for AWGN channel and between 28.02% and 65.16% for the fading channel, which means that at least 25 percent of error data-frames have been corrected with the help of JSCD. Take *AerialCity* with QP 22 in AWGN channel as an example, as shown in Fig. 5(a), the FER is 7.72% when $\frac{E_b}{N_0} = 2.0$ dB, and Δ_{FER} is 2.80% which means that the FER can be reduced to 7.72% − 2.80% = 4.92% after performing JSCD. Δ_{EC} is 36.20% which means that 36.20 percent of error data-frames have been corrected. Similarly, when $\frac{E_b}{N_0} = 2.1, 2.2, 2.3$ dB, the Δ_{FER}s are 1.05%, 0.53%, 0.20% respectively, and the Δ_{EC}s are 42.22%, 43.57%, 51.32% respectively. For QP 27, the FER improvements for $\frac{E_b}{N_0} = 2.0, 2.1, 2.2, 2.3$ dB are 2.57%, 0.86%, 0.50%, 0.27% respectively, and the correction ratios are 33.70%, 39.63%, 47.11% and 62.44% respectively. Results for QP 32 and 37 show similar performances.

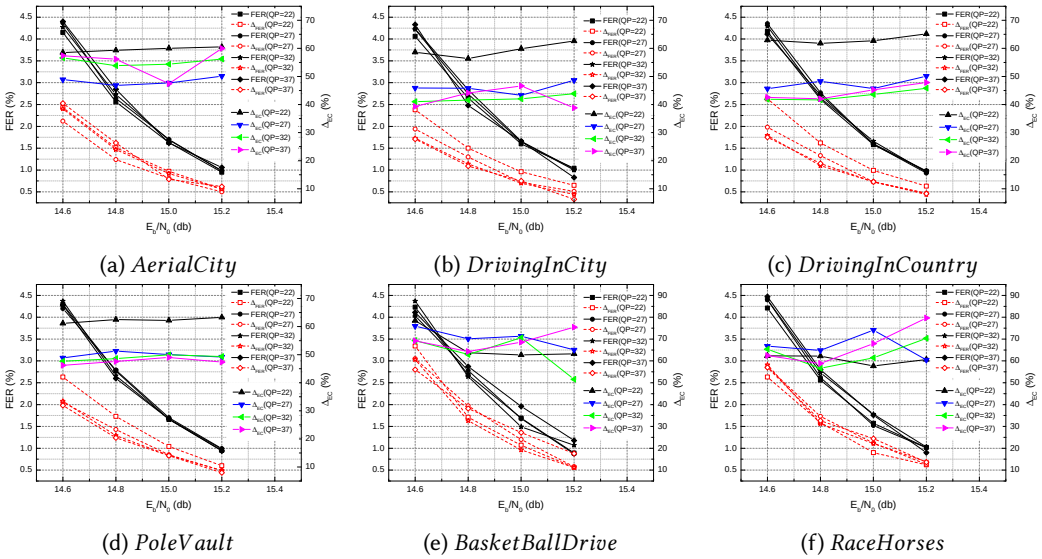
Generally, as $\frac{E_b}{N_0}$ grows, more and more percentage of data-frames can be corrected, while less FER improvements are obtained. In the perspective of QP, it seems that more error data-frames could be corrected for larger QPs. The best case for Δ_{FER} lies in *AerialCity* with $\frac{E_b}{N_0} = 2.3$ dB and QP=37, in that, 3.01% FER improvement has been achieved. The best case for Δ_{EC} lies in *BasketBallDrive*



(a) *AerialCity* (b) *DrivingInCity* (c) *DrivingInCountry*

(d) *PoleVault* (e) *BasketBallDrive* (f) *RaceHorses*

Fig. 5. The FER, Δ_{FER} and Δ_{EC} for the AWGN channel.



(a) *AerialCity* (b) *DrivingInCity* (c) *DrivingInCountry*

(d) *PoleVault* (e) *BasketBallDrive* (f) *RaceHorses*

Fig. 6. The FER, Δ_{FER} and Δ_{EC} for the Rayleigh fading channel.

with $\frac{E_b}{N_0}$ = 2.3 dB and QP=37, in that, 73.41% error data-frames have been corrected. Averagely, for the sequences with different QPs and $\frac{E_b}{N_0}$ levels, 1.07% FER improvement has been achieved and 44.09% error data-frames have been corrected.

The proposed approach also works well in the fading channel. As shown in Fig. 6, it has the similar performances as in the AWGN channel. The best case for Δ_{FER} lies in *DrivingInCountry* with $\frac{E_b}{N_0}$ = 14.6 dB and QP=22, in that, 2.63% FER improvement has been achieved. The best case for

Δ_{EC} also lies in *DrivingInCountry* with $\frac{E_b}{N_0}$ = 15.2 dB and QP=37, in that, 65.16% error data-frames have been corrected. On average, 1.11% FER improvement has been achieved and 46.98% of the error data-frames have been corrected. Overall, the scheme performs a little better in the fading channel than in the AWGN channel.

5.3 Video Quality Performance Evaluation

To evaluate the qualities of the 360° videos, Weighted-to-Spherically-uniform Peak Signal-to-Noise Ratio (WS-PSNR) and Voronoi-based Video Multimethod Assessment Fusion (VI-VMAF)[11] are utilized. On the other hand, PSNR and Structural SIMilarity (SSIM) are measured for the regular 2D videos. In this subsection, the proposed JSCD (denoted as "Pro.") is mainly compared to the conventional source-channel separated decoding scheme (denoted as "NoJ.") and the proposed JSCD scheme with channel decoded error bits' range given as ground truth (denoted as "Pro_RGT" and as "RGT" in the tables due to space limitation). In addition, the qualities of videos reconstructed from the original HEVC encoded bitstreams are also given as the baseline for comparisons (denoted as "Rec."). For a specific metric, if the values for "NoJ.", "Pro_RGT" and "RGT" are A, B and C respectively, then the performance improvements of "Pro_RGT" and "RGT" are calculated as $\Delta_{Pro} = (B - A)/A$ and $\Delta_{RGT} = (C - A)/A$, respectively. In the following tables, Δ_{Pro} and Δ_{RGT} are denoted as the performance improvements.

Tables 5 and 6 present the WS-PSNR and VI-VMAF results for the 360° videos in AWGN channel. It can be seen that "Pro." and "Pro_RGT" has improved all the video qualities significantly in all cases of $\frac{E_b}{N_0}$ compared to "NoJ.". Specifically, in terms of WS-PSNR, the best improvements lies in *DrivingInCity* with QP 27 for $\frac{E_b}{N_0}$ = 2.2 dB, which are $(25.52 - 17.29)/17.29 \times 100\% \approx 48\%$ for "Pro." and $(36.68 - 17.29)/17.29 \times 100\% \approx 110\%$ for "Pro_RGT", as shown in Table 5. The "Avg." and "All Avg." improvement values presented in the table are averaged from the four videos' data. As we can see, the overall average WS-PSNR improvements of "Pro." for QP 22, 27, 32, 37 are 22%, 25%, 24% and 20%, respectively. In terms of VI-VMAF, as shown in Table 6, the biggest improvements reach to $(81.48 - 70.75)/70.75 \times 100\% \approx 15\%$ for "Pro." for the *AerialCity* at $\frac{E_b}{N_0}$ = 2.3 dB with QP 27, and $(82.59 - 61.60)/61.60 \times 100\% \approx 34\%$ for "Pro_RGT" for the *DrivingInCity* at $\frac{E_b}{N_0}$ = 2.2 dB with QP 27. In average, the VI-VMAF improvements of "Pro." for QP 22, 27, 32, 37 are 6%, 7%, 8% and 7%, respectively.

Table 7 and 8 give the detailed PSNR and SSIM results for the 2D videos in AWGN channels. They also show that "Pro." and "Pro_RGT" work fine in improving 2D video qualities in all cases of $\frac{E_b}{N_0}$ compared to "NoJ.". Specifically, in terms of PSNR, the best improvement for "Pro." lies in *RaceHorses* with QP 37 for $\frac{E_b}{N_0}$ = 2.1 dB, which is $(21.37 - 16.31)/16.31 \times 100\% \approx 31\%$. The best improvement for "Pro_RGT" lies in *RaceHorses* with QP 32 for $\frac{E_b}{N_0}$ = 2.1 dB, which is $(27.30 - 15.02)/15.02 \times 100\% \approx 82\%$, as shown in Table 7. The overall average PSNR improvements of "Pro." for QP 22, 27, 32, 37 are 7%, 13%, 20% and 17%, respectively. In terms of SSIM, as shown in Table 8, the biggest improvements have reached to $(0.68 - 0.37)/0.37 \times 100\% \approx 84\%$ for "Pro." for *RaceHorses* at $\frac{E_b}{N_0}$ = 2.1 dB with QP 37 and $(0.63 - 0.15)/0.15 \times 100\% \approx 320\%$ for "Pro_RGT" at $\frac{E_b}{N_0}$ = 2.0 dB with QP 32. In average, the SSIM improvements of "Pro." for QP 22, 27, 32, 37 are 24%, 34%, 37% and 39%, respectively.

For the Rayleigh fading channels, the proposed algorithms achieve similar results. Table 9 and 10 show that all the metrics are improved in all cases of the $\frac{E_b}{N_0}$ and QPs for both 360° and regular 2D videos. Compared to "NoJ.", the overall average improvements of WS-PSNR and VI-VMAF for the 360° videos are 19% and 6% respectively, and the overall average improvements of PNSR and SSIM for the 2D videos are 15% and 37% respectively.

Table 5. Average WS-PSNR for 360° videos in AWGN channels.

Seq.	$\frac{E_b}{N_0}$ (dB)	QP=22 (dB)				QP=27 (dB)				QP=32 (dB)				QP=37 (dB)			
		NoJ.	Pro.	RGT	Rec.	NoJ.	Pro.	RGT	Rec.	NoJ.	Pro.	RGT	Rec.	NoJ.	Pro.	RGT	Rec.
A.C.	2.3	23.99	30.14	36.15	43.99	26.30	35.49	41.54	42.34	32.94	37.32	39.62	41.17	29.49	35.25	37.95	39.72
	2.2	16.44	21.20	28.17	43.99	20.50	27.95	34.87	42.34	26.47	32.62	37.68	41.17	26.78	33.08	36.20	39.72
	2.1	13.10	15.47	19.16	43.99	16.03	20.21	26.10	42.34	18.21	25.63	32.71	41.17	20.58	25.98	32.99	39.72
	2.0	10.86	11.75	15.45	43.99	12.95	13.77	17.45	42.34	14.73	18.02	24.26	41.17	16.25	18.87	26.23	39.72
D.I.C.	2.3	25.02	32.58	37.54	46.05	24.38	30.91	38.58	44.27	32.68	37.03	39.41	42.77	32.20	37.07	39.62	41.16
	2.2	18.16	24.21	34.86	46.05	17.29	**25.52**	**36.38**	44.27	22.74	29.90	37.50	42.77	21.50	30.09	36.49	41.16
	2.1	14.58	20.75	27.81	46.05	14.63	18.55	26.49	44.27	17.12	21.84	32.02	42.77	19.31	24.43	31.89	41.16
	2.0	12.61	14.17	17.97	46.05	11.91	13.68	18.08	44.27	14.75	17.75	23.88	42.77	14.37	15.01	20.94	41.16
D.I.Cnt.	2.3	23.00	28.88	38.57	46.12	26.21	31.63	38.40	43.99	31.10	37.99	39.79	42.36	29.98	36.11	38.47	40.74
	2.2	18.65	22.76	32.44	46.12	20.04	23.12	33.29	43.99	23.12	29.61	36.50	42.36	22.88	29.32	36.53	40.74
	2.1	16.74	19.25	25.44	46.12	14.76	19.17	28.49	43.99	17.37	22.81	31.54	42.36	17.46	19.45	26.81	40.74
	2.0	15.04	15.53	18.86	46.12	13.07	13.76	17.38	43.99	13.65	16.41	22.64	42.36	15.61	17.34	24.79	40.74
P.V.	2.3	18.62	22.74	32.69	42.06	20.80	29.75	37.61	39.39	28.22	32.59	35.86	37.80	24.94	30.41	33.21	36.34
	2.2	16.76	19.77	24.58	42.06	15.51	21.56	31.35	39.39	20.28	27.07	33.47	37.80	19.20	23.80	32.24	36.34
	2.1	11.22	14.41	16.38	42.06	13.01	15.77	22.47	39.39	15.30	20.64	27.42	37.80	16.68	20.07	28.02	36.34
	2.0	13.48	15.94	16.01	42.06	11.86	12.29	14.07	39.39	13.20	14.56	20.69	37.80	15.45	16.05	20.68	36.34
			Δ_{Pro}	Δ_{RGT}			Δ_{Pro}	Δ_{RGT}			Δ_{Pro}	Δ_{RGT}			Δ_{Pro}	Δ_{RGT}	
Avg.	2.3	-	**26%**	**60%**	-	-	**31%**	**60%**	-	-	**16%**	**24%**	-	-	**19%**	**28%**	-
	2.2	-	**26%**	**71%**	-	-	**34%**	**85%**	-	-	**29%**	**57%**	-	-	**29%**	**57%**	-
	2.1	-	**26%**	**60%**	-	-	**26%**	**77%**	-	-	**34%**	**82%**	-	-	**21%**	**62%**	-
	2.0	-	**10%**	**31%**	-	-	**7%**	**35%**	-	-	**18%**	**62%**	-	-	**9%**	**50%**	-
All Avg.	-	-	**22%**	**56%**	-	-	**25%**	**64%**	-	-	**24%**	**56%**	-	-	**20%**	**49%**	-

Table 6. Average VI-VMAF [11] for 360° videos in AWGN channels.

Seq.	$\frac{E_b}{N_0}$ (dB)	QP=22 (dB)				QP=27 (dB)				QP=32 (dB)				QP=37 (dB)			
		NoJ.	Pro.	RGT	Rec.	NoJ.	Pro.	RGT	Rec.	NoJ.	Pro.	RGT	Rec.	NoJ.	Pro.	RGT	Rec.
A.C.	2.3	68.27	75.06	82.3	92.66	70.75	**81.48**	89.31	90.4	78.36	83.78	86.76	88.82	74.31	81.18	84.59	86.89
	2.2	60.8	65.4	72.82	92.66	64.7	72.57	80.71	90.4	70.93	77.98	84.24	88.82	71.27	78.53	82.37	86.89
	2.1	57.80	59.91	63.38	92.66	60.42	64.41	70.53	90.4	62.47	70.02	78.08	88.82	64.78	70.4	78.42	86.89
	2.0	55.89	56.64	59.89	92.66	57.67	58.39	61.75	90.4	59.24	62.29	68.56	88.82	60.63	63.1	70.67	86.89
D.I.C.	2.3	69.37	77.93	84.07	95.55	68.68	75.96	85.40	93.05	78.05	83.42	86.49	90.98	77.48	83.47	86.76	88.80
	2.2	62.42	68.50	80.70	95.55	61.60	69.90	**82.59**	93.05	66.97	74.78	84.01	90.98	65.70	75.00	82.73	88.80
	2.1	59.11	64.95	72.41	95.55	59.15	62.79	70.96	93.05	61.44	66.05	77.26	90.98	63.53	68.74	77.11	88.80
	2.0	57.38	58.74	62.24	95.55	56.78	58.31	62.34	93.05	59.26	62.03	68.16	90.98	58.92	59.49	65.14	88.80
D.I.Cnt.	2.3	67.24	73.62	85.39	95.65	70.65	76.80	85.17	92.66	76.18	84.64	86.98	90.42	74.88	82.25	85.26	88.24
	2.2	62.89	66.99	77.76	95.65	64.25	67.36	78.78	92.66	67.36	74.45	82.74	90.42	67.11	74.12	82.78	88.24
	2.1	61.08	63.47	69.82	95.65	59.27	63.39	73.18	92.66	61.67	67.04	76.69	90.42	61.76	63.67	71.31	88.24
	2.0	59.52	59.97	63.09	95.65	57.77	58.38	61.68	92.66	58.28	60.77	66.87	90.42	60.04	61.64	69.12	88.24
P.V.	2.3	62.86	66.97	78.06	90.02	65.00	74.61	84.16	86.46	72.87	77.94	81.94	84.40	69.28	75.37	78.69	82.54
	2.2	61.10	63.98	68.90	90.02	59.95	65.76	76.47	86.46	64.48	71.59	79.00	84.40	63.42	68.07	77.52	82.54
	2.1	56.19	58.96	60.75	90.02	57.72	60.19	66.69	86.46	59.76	64.84	71.98	84.40	61.02	64.28	72.65	82.54
	2.0	58.13	60.34	60.41	90.02	56.73	57.1	58.65	86.46	57.89	59.09	64.89	84.40	59.89	60.44	64.88	82.54
			Δ_{Pro}	Δ_{RGT}			Δ_{Pro}	Δ_{RGT}			Δ_{Pro}	Δ_{RGT}			Δ_{Pro}	Δ_{RGT}	
Avg.	2.3	-	**10%**	**23%**	-	-	**12%**	**25%**	-	-	**8%**	**12%**	-	-	**9%**	**13%**	-
	2.2	-	**7%**	**21%**	-	-	**10%**	**27%**	-	-	**11%**	**22%**	-	-	**11%**	**22%**	-
	2.1	-	**6%**	**14%**	-	-	**6%**	**19%**	-	-	**9%**	**24%**	-	-	**6%**	**19%**	-
	2.0	-	**2%**	**6%**	-	-	**1%**	**7%**	-	-	**4%**	**14%**	-	-	**2%**	**13%**	-
All Avg.	-	-	**6%**	**16%**	-	-	**7%**	**19%**	-	-	**8%**	**18%**	-	-	**7%**	**17%**	-

The experimental results show that the proposed JSCD scheme is not sensitive to the video encoding QPs but to the channel noise levels. In addition, from the above tables, it can be seen that the proposed JSCD is inferior to "Pro_RGT", which means that higher video quality can be obtained if higher accuracy of the error bit range estimation is achieved. It indicates that the proposed

Table 7. Average PSNR for 2D videos in AWGN channels.

Seq.	$\frac{E_b}{N_0}$ (dB)	QP=22 (dB)				QP=27 (dB)				QP=32 (dB)				QP=37 (dB)			
		NoJ.	Pro.	RGT	Rec.	NoJ.	Pro.	RGT	Rec.	NoJ.	Pro.	RGT	Rec.	NoJ.	Pro.	RGT	Rec.
B.D.	2.3	18.96	21.89	28.13	39.76	20.94	27.09	31.47	38.17	25.03	30.98	34.31	36.41	24.85	29.76	31.86	34.40
	2.2	15.79	17.34	21.82	39.76	16.72	19.89	31.64	38.17	21.06	25.55	32.92	36.41	20.32	24.98	29.59	34.40
	2.1	13.68	14.62	17.03	39.76	14.85	17.33	23.98	38.17	15.27	18.62	25.51	36.41	17.20	20.19	25.32	34.40
	2.0	13.31	14.30	15.22	39.76	13.98	15.04	16.91	38.17	14.15	15.39	18.40	36.41	15.90	18.35	22.45	34.40
R.H.	2.3	21.75	24.61	30.19	39.32	27.27	32.00	34.72	35.19	21.09	24.75	27.80	32.03	27.77	28.20	28.20	29.20
	2.2	18.44	19.37	27.85	39.32	17.41	19.53	29.30	35.19	21.21	26.45	30.53	32.03	19.69	23.64	28.36	29.20
	2.1	13.59	13.90	21.33	39.32	15.97	16.01	24.60	35.19	15.02	19.60	**27.30**	32.03	16.31	**21.37**	26.14	29.20
	2.0	13.18	13.20	13.98	39.32	13.66	13.73	16.94	35.19	13.22	14.45	19.90	32.03	14.86	16.52	20.04	29.20
			Δ_{Pro}	Δ_{RGT}			Δ_{Pro}	Δ_{RGT}			Δ_{Pro}	Δ_{RGT}			Δ_{Pro}	Δ_{RGT}	
Avg.	2.3	-	**14%**	**43%**	-	-	**23%**	**37%**	-	-	**21%**	**35%**	-	-	**10%**	**14%**	-
	2.2	-	**7%**	**45%**	-	-	**15%**	**79%**	-	-	**23%**	**50%**	-	-	**22%**	**45%**	-
	2.1	-	**5%**	**41%**	-	-	**8%**	**58%**	-	-	**26%**	**74%**	-	-	**24%**	**54%**	-
	2	-	**4%**	**10%**	-	-	**4%**	**22%**	-	-	**9%**	**40%**	-	-	**13%**	**38%**	-
All Avg.	-	-	**7%**	**35%**	-	-	**13%**	**49%**	-	-	**20%**	**50%**	-	-	**17%**	**38%**	-

Table 8. Average SSIM for 2D videos in AWGN channels.

Seq.	$\frac{E_b}{N_0}$ (dB)	QP=22 (dB)				QP=27 (dB)				QP=32 (dB)				QP=37 (dB)			
		NoJ.	Pro.	RGT	Rec.	NoJ.	Pro.	RGT	Rec.	NoJ.	Pro.	RGT	Rec.	NoJ.	Pro.	RGT	Rec.
B.D.	2.3	0.54	0.71	0.88	0.99	0.66	0.85	0.92	0.99	0.79	0.92	0.97	0.98	0.79	0.89	0.93	0.97
	2.2	0.35	0.48	0.71	0.99	0.45	0.64	0.91	0.99	0.66	0.80	0.96	0.98	0.67	0.84	0.93	0.97
	2.1	0.19	0.30	0.47	0.99	0.30	0.51	0.75	0.99	0.43	0.61	0.80	0.98	0.45	0.63	0.80	0.97
	2.0	0.17	0.21	0.32	0.99	0.16	0.24	0.48	0.99	0.24	0.32	0.55	0.98	0.28	0.45	0.71	0.97
R.H.	2.3	0.69	0.78	0.87	0.99	0.81	0.93	0.98	0.99	0.65	0.77	0.87	0.98	0.87	0.89	0.89	0.96
	2.2	0.54	0.62	0.83	0.99	0.48	0.6	0.9	0.99	0.64	0.82	0.91	0.98	0.63	0.77	0.93	0.96
	2.1	0.25	0.32	0.66	0.99	0.35	0.46	0.78	0.99	0.34	0.58	0.81	0.98	0.37	**0.68**	0.85	0.96
	2.0	0.14	0.13	0.31	0.99	0.13	0.14	0.46	0.99	0.15	0.27	**0.63**	0.98	0.22	0.37	0.61	0.96
			Δ_{Pro}	Δ_{RGT}			Δ_{Pro}	Δ_{RGT}			Δ_{Pro}	Δ_{RGT}			Δ_{Pro}	Δ_{RGT}	
Avg.	2.3	-	**21%**	**42%**	-	-	**21%**	**29%**	-	-	**17%**	**28%**	-	-	**7%**	**10%**	-
	2.2	-	**24%**	**73%**	-	-	**33%**	**95%**	-	-	**25%**	**44%**	-	-	**24%**	**43%**	-
	2.1	-	**41%**	**157%**	-	-	**49%**	**135%**	-	-	**55%**	**109%**	-	-	**60%**	**101%**	-
	2.0	-	**10%**	**103%**	-	-	**31%**	**224%**	-	-	**51%**	**203%**	-	-	**64%**	**164%**	-
All Avg.	-	-	**24%**	**94%**	-	-	**34%**	**121%**	-	-	**37%**	**96%**	-	-	**39%**	**79%**	-

Table 9. Average WS-PSNR and VI-VMAF for 360° videos in Rayleigh fading channels.

Metric	$\frac{E_b}{N_0}$ (dB)	QP=22				QP=27				QP=32				QP=37			
		NoJ.	Pro.	RGT	Rec.	NoJ.	Pro.	RGT	Rec.	NoJ.	Pro.	RGT	Rec.	NoJ.	Pro.	RGT	Rec.
WS-PSNR (dB)	15.2	24.46	**29.53**	43.39	44.56	25.14	**30.52**	41.57	42.50	27.10	**31.93**	40.04	41.03	28.91	**32.88**	38.46	39.49
	15.0	20.72	**26.21**	42.70	44.56	20.10	**23.76**	40.08	42.50	23.07	**26.86**	39.34	41.03	26.21	**29.91**	38.23	39.49
	14.8	17.64	**20.71**	39.03	44.56	16.85	**20.81**	37.83	42.50	19.76	**23.59**	38.44	41.03	21.33	**25.64**	37.39	39.49
	14.6	14.76	**16.77**	35.80	44.56	13.92	**16.51**	35.62	42.50	17.51	**21.19**	36.28	41.03	19.34	**22.56**	35.42	39.49
VI-VMAF	15.2	68.77	**74.36**	91.83	93.45	69.49	**75.50**	89.35	90.61	71.63	**77.16**	87.31	88.63	73.65	**78.29**	85.25	86.59
	15.0	64.92	**70.65**	90.89	93.45	64.30	**68.03**	87.37	90.61	67.31	**71.36**	86.39	88.63	70.65	**74.80**	84.95	86.59
	14.8	61.93	**64.91**	85.99	93.45	61.18	**65.01**	84.44	90.61	63.97	**67.85**	85.22	88.63	65.53	**70.03**	83.87	86.59
	14.6	59.27	**61.11**	81.87	93.45	58.52	**60.87**	81.64	90.61	61.80	**65.39**	82.47	88.63	63.56	**66.78**	81.39	86.59

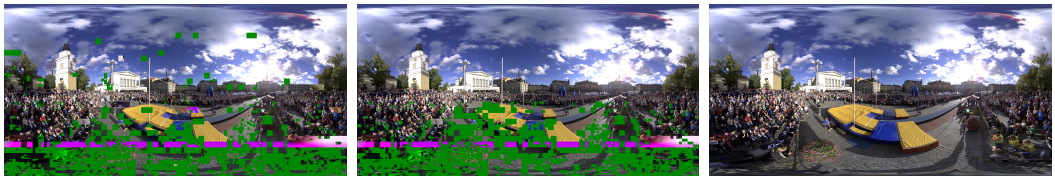
JSCD could be further improved. The possible direction would be HEVC semantic and syntax error checking, so as to improve the accuracy of error bit range estimation.

Fig. 7 and 8 give the visual results of the final recovered 360° videos in EquiRectangular (ERP) and CubeMap (CMP) projections respectively, and Fig. 9 gives those of the two 2D videos. Obviously, the proposed scheme can successfully recover several parts of the slices inside the video frames, thus improving the whole video's quality. It can be observed that, for some video frames, there are gaps between "Pro" and "Pro_RGT" in terms of percentages of recovered slices, which proves that the accuracy of error bit range prediction plays a key role in improving video quality.

Table 10. Average PSNR and SSIM for 2D videos in Rayleigh fading channels.

Metric	$\frac{E_b}{N_0}$ (dB)	QP=22				QP=27				QP=32				QP=37			
		NoJ.	Pro.	RGT	Rec.	NoJ.	Pro.	RGT	Rec.	NoJ.	Pro.	RGT	Rec.	NoJ.	Pro.	RGT	Rec.
PSNR (dB)	15.2	17.33	**21.87**	37.09	39.54	18.77	**21.01**	34.54	36.68	21.47	**24.40**	33.91	34.22	21.94	**24.72**	31.11	31.80
	15.0	16.02	**19.09**	29.5	39.54	15.89	**19.04**	32.89	36.68	17.33	**20.36**	33.03	34.22	19.48	**21.71**	29.47	31.80
	14.8	14.86	**17.20**	30.38	39.54	14.84	**17.72**	31.97	36.68	16.52	**20.06**	31.41	34.22	17.46	**19.76**	30.59	31.80
	14.6	13.86	**15.96**	23.43	39.54	14.17	**16.08**	25.44	36.68	15.42	**16.78**	30.66	34.22	16.95	**18.29**	29.40	31.80
SSIM	15.2	0.50	**0.68**	0.98	0.99	0.54	**0.64**	0.97	0.99	0.67	**0.76**	0.98	0.98	0.67	**0.77**	0.95	0.97
	15.0	0.35	**0.55**	0.88	0.99	0.37	**0.56**	0.93	0.99	0.49	**0.64**	0.97	0.98	0.60	**0.68**	0.92	0.97
	14.8	0.27	**0.43**	0.90	0.99	0.33	**0.48**	0.93	0.99	0.42	**0.61**	0.95	0.98	0.48	**0.61**	0.94	0.97
	14.6	0.21	**0.36**	0.74	0.99	0.25	**0.38**	0.82	0.99	0.32	**0.41**	0.93	0.98	0.38	**0.48**	0.92	0.97

(a) PSNR: 18.17 dB, SSIM: 0.74 (b) PSNR: 23.28 dB, SSIM: 0.91 (c) PSNR: 36.60 dB, SSIM: 0.98



(d) PSNR: 15.41 dB, SSIM: 0.66 (e) PSNR: 16.33 dB, SSIM: 0.70 (f) PSNR: 35.11 dB, SSIM: 0.98

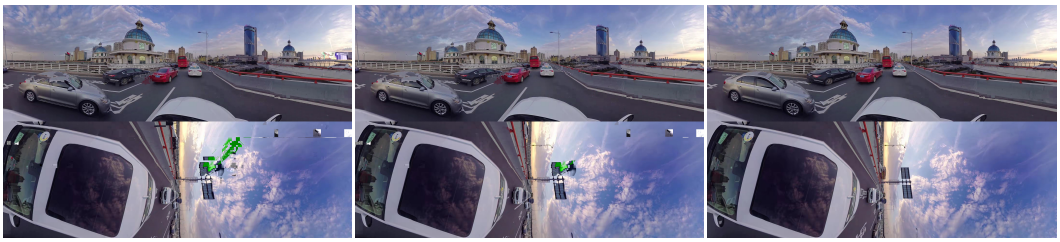
Fig. 7. Visual results of 360° videos in ERP format. (a), (b), (c), (d), (e), (f) are from "NoJ.", "Pro.", "Pro_RGT" schemes for *DrivingInCountry* and *PoleVault* video sequences, respectively.(QP=32, AWGN channel, $\frac{E_b}{N_0} = 2.1$ dB)

5.4 Complexity Analysis

Fig. 4 shows that there are iterative loops involved in the proposed JSCD scheme, which costs extra computations depending on the number of loops in performing iterative JSCD. In the iterative JSCD process, the R-SCFlip decoding for data-frames with error bits needs to be run recurrently and the HEVC decoding for the corresponding NAL units needs to be run recurrently as well. Therefore, the increased computational complexity can be measured by the increased number of JSCD iterations.

The experimental results are collected from the 360° videos streaming in Rayleigh fading channels. Table 11 gives the average percentage of increased iterative decoding loops when applying JSCD process, where η_c is the percentage of channel data-frames needed to do extra R-SCFlip decoding and η_s is the percentage of NAL units need to do extra HEVC decoding. The overall increased computational complexity can be measured by $\eta_c + \eta_s$. For lower $\frac{E_b}{N_0}$, more data-frames tend to be channel decoded incorrectly, and more iterative decoding loops are involved when performing JSCD. In this case, more extra computations are required. Each loop involves one data-frame channel decoding process and one NAL unit source decoding process. Each experiment with the same configuration is run for 100 times. The number of iterative decoding loops is recorded and the average value is computed. It is found that the worst-case happens when $QP = 27$ and $\frac{E_b}{N_0} = 14.6$ dB, where the proposed JSCD has to run averagely about 85.97 times extra decoding loops. In this case, the average total number of data-frames and NAL units are 1984 and 8932 (according to

(a) PSNR: 19.97 db, SSIM: 0.74 (b) PSNR: 28.52 dB, SSIM: 0.95 (c) PSNR: 37.00 dB, SSIM: 0.97



(d) PSNR: 24.80 dB, SSIM: 0.91 (e) PSNR: 25.98 dB, SSIM: 0.92 (f) PSNR: 36.35 dB, SSIM: 0.97

Fig. 8. Visual results of CMP projected 360° videos. (a), (b), (c), (d), (e), (f) are from "NoJ.", "Pro.", "Pro_RGT" schemes for *AerialCity* and *DrivingInCity* sequences, respectively.(QP=32, AWGN channel, $\frac{E_b}{N_0}$ = 2.2 dB)

(a) PSNR: 22.91 dB, SSIM: 0.78 (b) PSNR: 27.17 dB, SSIM: 0.94 (c) PSNR: 37.70 dB, SSIM: 0.99

(d) PSNR: 20.43 dB, SSIM: 0.82 (e) PSNR: 28.62 dB, SSIM: 0.96 (f) PSNR: 31.66 dB, SSIM: 0.98

Fig. 9. Visual results of 2D videos. (a), (b), (c), (d), (e), (f) are "NoJ.", "Pro.", "Pro_RGT" schemes for *BasketBallDrive* (QP=32, AWGN channel, $\frac{E_b}{N_0}$ = 2.2 dB) and *RaceHorses* video sequences (QP=32, Fading channel, $\frac{E_b}{N_0}$ = 15.0 dB), respectively.

the parameter configuration given in Table 4), respectively. Thus, the ratio of extra computation overhead for channel and HEVC decoding is 4.35% and 0.94%, respectively, as specified in Table 11, and the total computation overhead is 4.35% + 0.94% = 5.29%. According to Table 11, the overall average computation overhead of $\eta_c + \eta_s$ is $(2.85\% + 3.01\% + 2.95\% + 2.95\%)/4 = 2.94\%$, which brings in 19% and 6% video quality improvements for WS-PSNR and VI-VMAF respectively, according to

the data from Table 9. Overall, the computational complexity results indicate that it is more worthy to perform JSCD for larger $\frac{E_b}{N_0}$.

Table 11. Average computational overheads. (Rayleigh fading channel)

$\frac{E_b}{N_0}$ (dB)	QP=22 (%)			QP=27 (%)			QP=32 (%)			QP=37 (%)		
	η_c	η_s	$\eta_c + \eta_s$	η_c	η_s	$\eta_c + \eta_s$	η_c	η_s	$\eta_c + \eta_s$	η_c	η_s	$\eta_c + \eta_s$
15.2	0.90	0.29	**1.19**	1.04	0.22	**1.26**	1.05	0.20	**1.25**	1.10	0.18	**1.28**
15.0	1.50	0.48	**1.98**	1.72	0.37	**2.06**	1.74	0.33	**2.07**	1.80	0.30	**2.10**
14.8	2.44	0.78	**3.22**	2.78	0.60	**3.38**	2.77	0.53	**3.30**	2.80	0.46	**3.26**
14.6	3.78	1.21	**4.99**	4.35	0.94	**5.29**	4.36	0.83	**5.19**	4.44	0.73	**5.17**
Avg.			**2.85**			**3.01**			**2.95**			**2.95**

5.5 Comparison with Other JSCD Methodologies

The JSCD scheme in [41] (referred as "R. Perera *et al.* JSCD") is selected for comparison, which is compared with the "Pro_RGT" approach in this paper for fairness. In the experiments of "R. Perera *et al.* JSCD", a total of 60 data-frames (transport blocks) from *Foreman* video sequence are considered. Numbers of error blocks that can be recovered are $60 \times (20\% - 3\%) = 10.2$ and $60 \times (53\% - 22\%) = 18.6$ for SNR (measured in $\frac{E_s}{N_0}$) 11.5dB and 11.4dB, respectively. Thus, the improvements are $10.2/(60 \times 20\%) = 85.00\%$ and $18.6/(60 \times 53\%) = 58.49\%$, respectively. For comparison, in our experiments, the total number of transmission data-frames is 853, which is derived from 360° video sequence *AerialCity* with QP 32 in ERP projection. The average numbers of error decoded frames are 43.51 and 52.56 under fading channels with SNR=11.5dB and 11.4dB, respectively. "Pro_RGT" can correct 40.22 and 47.72 of them, respectively. Thus, the improvements are 92.44% and 90.79%, respectively. Table 12 summarizes the comparison result. Obviously, our "Pro_RGT" scheme shows better performance in terms of improvements in error data-frame recovery. It proves that, compared to turbo decoders, the polar decoder can correct more data-frames when the extrinsic information of positions of error bits is given.

Table 12. Comparisons of improvements of error data-frame recovery.

SNR (dB)	R. Perera *et al.* JSCD in [41]	Pro_RGT
11.5	85.00%	92.44%
11.4	58.49%	90.79%

In terms of the quality of visual experience, "R. Perera *et al.* JSCD" has given the SSIM results for channel SNR from 11.375dB to 11.6dB in [41]. The best case turned out to be the *Beergarden* video sequence with higher resolution. Therefore, only comparisons with this case are made here. The experiments for "Pro_RGT" are also performed in Rayleigh fading channel with the corresponding SNRs for the *AerialCity* in ERP. Fig. 10 gives the comparison. It shows that "Pro_RGT" produces higher SSIM values in general, and significantly performs better in the small SNR range. The largest performance gap lies in SNR=11.375dB, which equals about 25% improvement. These improvements are owning to the higher ratio of error data-frame recovery provided by the polar decoder utilizing HEVC semantic and syntax error checking.

6 CONCLUSIONS

We propose a novel Joint Source-Channel Decoding (JSCD) approach of polar codes for High-Efficiency Video Coding (HEVC) based video streaming. According to the semantic and syntax

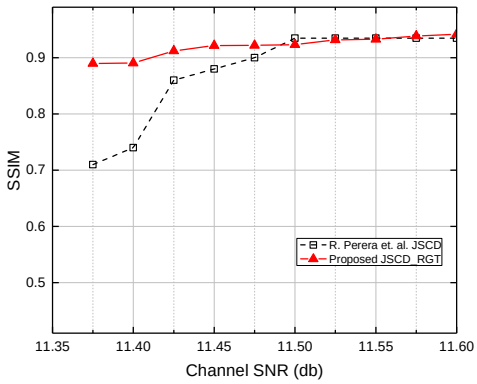


Fig. 10. Comparisons of end user viewing experience.

errors reported by the HEVC decoder, the error bit positions in the input video bitstreams are estimated by a Kernel Density Estimation (KDE) fitting approach. By using the estimated error bit position ranges, an R-SCFlip polar decoding algorithm is presented. By combining the KDE error bit range estimator and the R-SCFlip polar decoder together, an iterative JSCD methodology is proposed. Experimental results show that the proposed scheme demonstrates significant performance improvements compared to the scheme without JSCD. Averagely, 1.09% Frame Error Ratio (FER) improvements have been achieved. The average Peak Signal-to-Noise Ratio (PSNR) and Weighted-to-Spherically-uniform PSNR (WS-PSNR) gains reach 14% and 21% for 2D and 360° videos, respectively. Experiments also indicate that the computational complexities paid for these improvements are affordable. Compared with benchmark JSCD methods, the proposed JSCD outperforms in recovering error data-frames, especially for small channel Signal-to-Noise Ratios (SNR).

REFERENCES

[1] 3GPP. 2018. *Multiplexing and channel coding.* Technical Specification (TS) 38.212. 3rd Generation Partnership Project (3GPP). version 15.2.0.

[2] Marwa Ben Abdessalem, Amin Zribi, Tadashi Matsumoto, Elsa Dupraz, and Ammar Bouallègue. 2020. LDPC-based joint source channel coding and decoding strategies for single relay cooperative communications. *Physical Communication* 38 (2020), 100947. https://doi.org/10.1016/j.phycom.2019.100947

[3] Orion Afisiadis, Alexios Balatsoukas-Stimming, and Andreas Burg. 2014. A low-complexity improved successive cancellation decoder for polar codes. In *2014 48th Asilomar Conference on Signals, Systems and Computers*. IEEE, 2116–2120. https://doi.org/10.1109/ACSSC.2014.7094848

[4] Erdal Arikan. 2009. Channel polarization: A method for constructing capacity-achieving codes for symmetric binary-input memoryless channels. *IEEE Transactions on Information Theory* 55, 7 (2009), 3051–3073. https://doi.org/10.1109/TIT.2009.2021379

[5] Yogesh Beeharry, Tulsi P Fowdur, and Krishnaraj MS Soyjaudah. 2019. Performance of hybrid binary and non-binary turbo decoding schemes for LTE and DVB-RCS standards. *ECTI Transactions on Electrical Engineering, Electronics, and Communications* 17, 1 (2019), 1–13. https://doi.org/10.37936/ecti-eec.2019171.215363

[6] Frank Bossen, Davin Flynn, and Karsten Suhring. 2013. HM software manual. *Document: JCTVC-M1010* (2013).

[7] Eirina Bourtsoulatze, David Burth Kurka, and Deniz Gündüz. 2019. Deep joint source-channel coding for wireless image transmission. *IEEE Transactions on Cognitive Communications and Networking* 5, 3 (2019), 567–579. https://doi.org/10.1109/TCCN.2019.2919300

[8] Adrien Cassagne, Olivier Hartmann, Mathieu Leonardon, Kun He, Camille Leroux, Romain Tajan, Olivier Aumage, Denis Barthou, Thibaud Tonnellier, Vincent Pignoly, et al. 2019. AFF3CT: A fast forward error correction toolbox! *SoftwareX* 10 (2019), 100345. https://doi.org/10.1016/j.softx.2019.100345

[9] Qiwang Chen, Lin Wang, Pingping Chen, and Guanrong Chen. 2019. Optimization of component elements in integrated coding systems for green communications: A survey. *IEEE Communications Surveys & Tutorials* 21, 3 (2019), 2977–2999. https://doi.org/10.1109/COMST.2019.2894154

[10] Stefania Colonnese, Francesca Cuomo, Luca Chiaraviglio, Valentina Salvatore, Tommaso Melodia, and Izhak Rubin. 2017. CLEVER: A cooperative and cross-layer approach to video streaming in HetNets. *IEEE Transactions on Mobile Computing* 17, 7 (2017), 1497–1510. https://doi.org/10.1109/TMC.2017.2774298

[11] Simone Croci, Cagri Ozcinar, Emin Zerman, Sebastian Knorr, Julián Cabrera, and Aljosa Smolic. 2020. Visual attention-aware quality estimation framework for omnidirectional video using spherical Voronoi diagram. *Quality and User Experience* 5, 1 (2020), 1–17.

[12] Pierre Duhamel and Michel Kieffer. 2009. Chapter 2 - Why joint source and channel decoding? In *Joint source-channel decoding: A cross-layer perspective with applications in video broadcasting*. Academic Press, 13–30.

[13] Pierre Duhamel and Michel Kieffer. 2009. *Joint source-channel decoding: A cross-layer perspective with applications in video broadcasting*. Academic Press.

[14] Sorina Dumitrescu. 2010. Fast joint source-channel decoding of convolutional coded Markov sequences with Monge property. *IEEE Transactions on Communications* 58, 1 (2010), 128–135. https://doi.org/10.1109/TCOMM.2010.01.080091

[15] Vassiliy A Epanechnikov. 1969. Non-parametric estimation of a multivariate probability density. *Theory of Probability & Its Applications* 14, 1 (1969), 153–158. https://doi.org/10.1137/1114019

[16] Furkan Ercan, Carlo Condo, and Warren J Gross. 2018. Improved bit-flipping algorithm for successive cancellation decoding of polar codes. *IEEE Transactions on Communications* 67, 1 (2018), 61–72. https://doi.org/10.1109/TCOMM.2018.2873322

[17] Shu Fan and Honglin Zhao. 2018. Delay-based cross-layer QoS scheme for video streaming in wireless ad hoc networks. *China Communications* 15, 9 (2018), 215–234. https://doi.org/10.1109/CC.2018.8456464

[18] Maria Fresia, Fernando Peréz-Cruz, H. Vincent Poor, and Sergio Verdú. 2010. Joint Source and Channel Coding. *IEEE Signal Processing Magazine* 27, 6 (2010), 104–113. https://doi.org/10.1109/MSP.2010.938080

[19] Shuping Gong, Liang Li, Ju Bin Song, and Husheng Li. 2017. Joint channel decoding and state estimation in cyber-physical systems. *IEEE Transactions on Wireless Communications* 16, 11 (2017), 7560–7573. https://doi.org/10.1109/TWC.2017.2750688

[20] 5G PPP Architecture Working Group et al. 2016. View on 5G architecture. *White Paper, July* (2016).

[21] Ammar Hadi, Emad Alsusa, and Arafat Al-Dweik. 2018. Information unequal error protection using polar codes. *IET Communications* 12, 8 (2018), 956–961. https://doi.org/10.1049/iet-com.2017.1195

[22] L Hanzo et al. 2011. Near-capacity H.264 multimedia communications using iterative joint source-channel decoding. *IEEE Communications Surveys & Tutorials* 14, 2 (2011), 538–564. https://doi.org/10.1109/SURV.2011.032211.00118

[23] Yuwen He, Xiaoyu Xiu, Yan Ye, et al. 2017. 360Lib software manual. *Joint Video Exploration Team (JVET) of ITU-T SG* 16 (2017).

[24] Xiem HoangVan and Byeungwoo Jeon. 2018. Joint layer prediction for improving SHVC compression performance and error concealment. *IEEE Transactions on Broadcasting* 65, 3 (2018), 504–520. https://doi.org/10.1109/TBC.2018.2881355

[25] Yongkai Huo, Chuan Zhu, and Lajos Hanzo. 2013. Spatio-temporal iterative source-channel decoding aided video transmission. *IEEE Transactions on vehicular technology* 62, 4 (2013), 1597–1609. https://doi.org/10.1109/TVT.2012.2227072

[26] Liqiang Jin and Hongwen Yang. 2018. Joint source-channel polarization with side information. *IEEE Access* 6 (2018), 7340–7349. https://doi.org/10.1109/ACCESS.2017.2788887

[27] Liqiang Jin, Pei Yang, and Hongwen Yang. 2018. Distributed joint source-channel decoding using systematic polar codes. *IEEE Communications Letters* 22, 1 (2018), 49–52. https://doi.org/10.1109/LCOMM.2017.2768036

[28] Mohamad Khas, Hamid Saeedi, and Reza Asvadi. 2018. Design and analysis of LDPC codes for joint source-channel decoding of two correlated sensors. *IET Communications* 12, 8 (2018), 1003–1010. https://doi.org/10.1049/iet-com.2017.1084

[29] Hasan Ali Khattak, Zoobia Ameer, Ud Ikram Din, and Muhammad Khurram Khan. 2019. Cross-layer design and optimization techniques in wireless multimedia sensor networks for smart cities. *Computer Science and Information Systems* 16, 1 (2019), 1–17. https://doi.org/10.2298/CSIS181115004K

[30] Hossein Kourkchi, William E Lynch, and M Omair Ahmad. 2018. A joint source channel arithmetic MAP decoder using probabilistic relations among intra modes in predictive video compression. In *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 1768–1772. https://doi.org/10.1109/ICASSP.2018.8462054

[31] David Burth Kurka and Deniz Gündüz. 2020. DeepJSCC-f: Deep joint source-channel coding of images with feedback. *IEEE Journal on Selected Areas in Information Theory* (2020). https://doi.org/10.1109/JSAIT.2020.2987203

[32] David Levine, William E Lynch, and Tho Le-Ngoc. 2010. Iterative joint source-channel decoding of H.264 compressed video. *Signal Processing: Image Communication* 25, 2 (2010), 75–87. https://doi.org/10.1016/j.image.2009.12.006

[33] Peihao Li, Fengbao Yang, Jing Zhang, Yun Guan, Anhong Wang, and Jie Liang. 2020. Synthesis-distortion-aware hybrid digital analog transmission for 3D videos. *IEEE Access* (2020). https://doi.org/10.1109/ACCESS.2020.2990198

[34] Jinzhi Lin, Shengzhong Feng, Zhile Yang, Yun Zhang, and Yong Zhang. 2020. A Novel Deep Neural Network Based Approach for Sparse Code Multiple Access. *Neurocomputing* 382 (2020), 52–63. https://doi.org/10.1016/j.neucom.2019.

11.066

[35] Zhi Liu, Susumu Ishihara, Ying Cui, Yusheng Ji, and Yoshiaki Tanaka. 2018. JET: Joint source and channel coding for error resilient virtual reality video wireless transmission. *Signal Processing* 147 (2018), 154–162. https://doi.org/10.1016/j.sigpro.2018.01.009

[36] Lei Luo, Taihai Yang, Ce Zhu, Zhi Jin, and Shu Tang. 2019. Joint texture/depth power allocation for 3-D video SoftCast. *IEEE Transactions on Multimedia* 21, 12 (2019), 2973–2984. https://doi.org/10.1109/TMM.2019.2919474

[37] Duc V Nguyen, Huyen TT Tran, and Truong Cong Thang. 2020. An evaluation of tile selection methods for viewport-adaptive streaming of 360-degree video. *ACM Transactions on Multimedia Computing, Communications, and Applications* 16, 1 (2020), 1–24. https://doi.org/10.1145/3373359

[38] Nguyen Quang Nguyen, William E Lynch, and Tho Le-Ngoc. 2010. Iterative joint source-channel decoding for H.264 video transmission using virtual checking method at source decoder. In *CCECE 2010*. IEEE, 1–4. https://doi.org/10.1109/CCECE.2010.5575234

[39] JCT-VC of ITU-T SG16 WP3 and ISO/IEC JTC1/SC29/WG11. 2020. *HEVC test model (HM)*. https://hevc.hhi.fraunhofer.de/HM-doc/

[40] Jounsup Park, Jenq-Neng Hwang, and Hung-Yu Wei. 2018. Cross-layer optimization for VR video multicast systems. In *2018 IEEE Global Communications Conference (GLOBECOM)*. IEEE, 206–212. https://doi.org/10.1109/GLOCOM.2018.8647389

[41] Ryan Perera, Hemantha Kodikara Arachchi, Muhammad Ali Imran, and Pei Xiao. 2016. Extrinsic information modification in the turbo decoder by exploiting source redundancies for HEVC video transmitted over a mobile channel. *IEEE Access* 4 (2016), 7186–7198. https://doi.org/10.1109/ACCESS.2016.2619259

[42] Cristina Perfecto, Mohammed S Elbamby, Javier Del Ser, and Mehdi Bennis. 2020. Taming the latency in multi-user VR 360°: A QoE-aware deep learning-aided multicast framework. *IEEE Transactions on Communications* 68, 4 (2020), 2491–2508. https://doi.org/10.1109/TCOMM.2020.2965527

[43] David W Scott. 2015. *Multivariate density estimation: theory, practice, and visualization*. John Wiley & Sons. https://doi.org/10.1007/978-3-642-21551-3_19

[44] ITU Telecommunication Standardization Sector. 2019. Recommendation ITU-T H.265: High efficiency video coding. *Ge-neva, Switzerland: Telecommunication Standardization Sector* (2019).

[45] Claude E Shannon. 1948. A mathematical theory of communication. *Bell system technical journal* 27, 3 (1948), 379–423. https://doi.org/10.1002/j.1538-7305.1948.tb01338.x

[46] Ying Wang, Minghai Qin, Krishna R Narayanan, Anxiao Jiang, and Zvonimir Bandic. 2016. Joint source-channel decoding of polar codes for language-based sources. In *2016 IEEE Global Communications Conference (GLOBECOM)*. IEEE, 1–6. https://doi.org/10.1109/GLOCOM.2016.7841934

[47] Yue Wang and Songyu Yu. 2005. Joint source-channel decoding for H.264 coded video stream. *IEEE Transactions on Consumer Electronics* 51, 4 (2005), 1273–1276. https://doi.org/10.1109/TCE.2005.1561855

[48] Yoshito Watanabe and Hideki Ochiai. 2016. A novel design and modeling of UEP-based compressed video broadcasting with multilevel coded modulation. *IEEE Transactions on Broadcasting* 62, 3 (2016), 598–609. https://doi.org/10.1109/TBC.2016.2576599

[49] Lin Xiang, Derrick Wing Kwan Ng, Toufiqul Islam, Robert Schober, Vincent WS Wong, and Jiaheng Wang. 2017. Cross-layer optimization of fast video delivery in cache-and buffer-enabled relaying networks. *IEEE Transactions on Vehicular Technology* 66, 12 (2017), 11366–11382. https://doi.org/10.1109/TVT.2017.2720481

[50] Zheng Yuan and Xinchen Zhao. 2012. Introduction of forward error correction and its application. In *2012 2nd International Conference on Consumer Electronics, Communications and Networks (CECNet)*. IEEE, 3288–3291.

[51] Alireza Zare, Maryam Homayouni, Alireza Aminlou, Miska M Hannuksela, and Moncef Gabbouj. 2019. 6K and 8K effective resolution with 4K HEVC decoding capability for 360 video streaming. *ACM Transactions on Multimedia Computing, Communications, and Applications* 15, 2s (2019), 1–22. https://doi.org/10.1145/3335053

[52] Xinglei Zhu and Chang W Chen. 2012. A joint layered scheme for reliable and secure mobile JPEG-2000 streaming. *ACM Transactions on Multimedia Computing, Communications, and Applications* 8, 3 (2012), 1–23. https://doi.org/10.1145/2240136.2240143

[53] Michael Zink, Ramesh Sitaraman, and Klara Nahrstedt. 2019. Scalable 360° video stream delivery: Challenges, solutions, and opportunities. *Proc. IEEE* 107, 4 (2019), 639–650. https://doi.org/10.1109/JPROC.2019.2894817