TRANSFORM COEFFICIENT CODING FOR SCREEN CONTENT IN VERSATILE VIDEO CODING (VVC)

Mohsen Abdoli[⋆†] *Félix Henry*[⋆] *Patrice Brault*[†] *Frédéric Dufaux* [†] *Pierre Duhamel*[†]

[⋆] Orange Labs, Cesson Sévigné, France
[†]L2S, CNRS - CentraleSupelec - Université Paris-Sud, Gif-sur-Yvette, France

ABSTRACT

A transform coefficient coding scheme is proposed for 4×4 blocks in Versatile Video Coding (VVC), targeting screen content applications. The proposed algorithm, called Unary Bitplane Coding (UBC), uses unary codes of the coefficient amplitudes and represents each block by their bitplanes. This representation allows exploiting further contextual information for source separation during the entropy coding. Experiments in the Joint Exploration test Model (JEM) show that replacing the existing transform coding with UBC only for 4×4 blocks brings on average 2.8% and 3.4% BD-R gain in the random access and all intra modes, respectively.

***Index Terms*—** Residual transform coding, VVC, Screen content

1. INTRODUCTION

With the growing applications of screen content in the past years, their efficient compression has become a priority for video codecs. In the context of future video coding, this priority is reflected in the "versatility" aspect of the future standard which is referred to as Versatile Video Coding (VVC) [1]. While targetting natural content is a priority, this codec is also supposed to have tools dedicated to screen content coding [2–7]. This is in contrast with High Efficiency Video Coding (HEVC), where an ad-hoc extension of the standard (i.e. HEVC-SCC) was dedicated to screen content [8].

Modern video coding standards benefit from block prediction tools to decorrelate the signal and remove its redundancies. A prediction phase is often followed by a residual coding phase to reconstruct a high quality image. Although the state-of-the-art prediction tools perform significantly well, a relatively large redundancy usually remains in the residual signal. In order to further compress the residual, this redundancy is exploited by residual transformation into the frequency domain. This allows transmission of residual information in a few transform coefficients (e.g. DCT, DST etc.).

The existing transform coefficient transmission algorithms of VVC, referred to as residual coding in the rest of this paper, has remained almost unchanged since HEVC [9]. For 4×4 blocks, that are solely considered in this paper, the following steps are performed: 1) signaling scan position of the last significant (i.e. non-zero) coefficient, 2) coding all amplitudes from the beginning until the last scan position, and finally 3) coding coefficient signs. For each amplitude, a prefix and a suffix is coded. The prefix uses a unary code of length up to three, depending to the amplitude value. If the amplitude is greater than or equal to two, the suffix for the remaining of the amplitude is coded in the bypass mode using Golomb-Rice and Exp-Golomb codes [9].

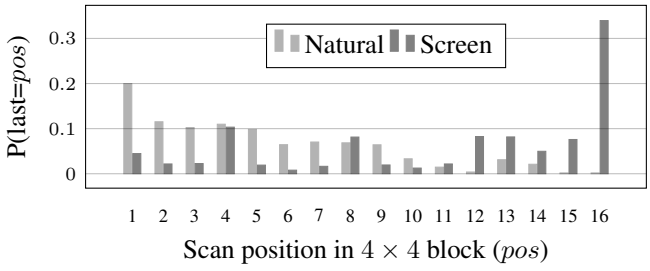


Fig. 1. Histogram of the last significant position in 4×4 blocks of natural and screen content.

Although the above algorithm is usually efficient in removing the correlations remaining in the transform blocks, it still has rooms to improve for screen content. For instance, Fig. 1 shows two histograms corresponding the last significant position in all 4×4 blocks, in natural and screen contents. As can be seen, this position tends to occur later in screen blocks than in natural blocks, leaving fewer non-significant coefficients at the end of blocks. To address this kind of problems, two major solutions have been proposed during the standardization of HEVC. The first solution is the Transform Skip Mode (TSM), which encodes residual values in the pixel domain [10]. The second approach, called Residual DPCM (RDPCM), performs a DPCM-based prediction on the pixel-domain residual obtained from the TSM [11–13].

Previously, it was shown that coding 4×4 blocks in VVC requires special attention as they are the smallest partitions and usually correspond to high detail textures [14]. Moreover, they are usually the most popular block size in all QP

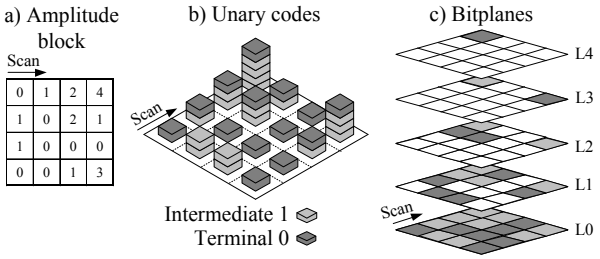


Fig. 2. Unary bitplane binarization of 4×4 amplitude blocks.

values [15]. Therefore, a transform coefficient coding, called Unary Bitplane Coding (UBC) is proposed in this paper for 4×4 blocks of screen content. In this algorithm, amplitudes are individually binarized using unary codes and then are represented with bitplanes. This representation allows to extract contextual information from coded bins and to categorize them with respect to their statistics. One Context-Adaptive Binary Arithmetic Coding (CABAC) model is then dedicated to each category during the coding. The rest of this paper is organized as follows. In section II, the amplitude coding framework is introduced. Section III describes the proposed configuration of this framework integrated in VVC. In section IV, experimental results are presented and finally, section V draws conclusions.

2. UBC FRAMEWORK

In this section, a flexible framework is described, aiming at obtaining as much contextual information as possible from the coded bins. Then, it optimizes a merging scheme to define a limited set of entropy coding contexts. From this aspect, the framework of this section shares its general idea with another residual coding algorithm proposed for the AV1 standard [16]

2.1. Binarization

The binarization scheme in UBC uses unary codes to produce bins of individual amplitude. The bins of amplitudes in a block are then represented by bitplanes as shown in Fig. 2, where each unsigned decimal value C in the amplitude block is first binarized to a unary code of $C + 1$ bins, including C intermediate '1' bins followed by a terminal '0' bin.

The main benefit of the unary bitplanes compared to binary bitplanes is the contextual information accessibility. For instance, let us assume that the encoder is currently in the *L*-th bitplane and bin values of all coefficients in the lower layers are available. For encoding a bin at the level *L* of a certain coefficient, the encoder attempts to access contextual information from neighboring amplitudes, available up to the level *L*. By accessing a bin value of '1' or '0' at the (*L-1*)-th level of one of its neighbors, the encoder can immediately decide whether the amplitude of the neighbor is less than *L-1*, or at least *L-1*. On the contrary, with binary codes, the encoder has to access all the bin values down to layer zero, in order to obtain the same contextual information about that neighbor.

Table 1. Description of four configurations *u, f, d* and *fd*, using F and D features, along with their corresponding entropy of the significance bin B.

cfg	Contextual features	Number of situations	P_{cfg}	H_{cfg}
u	None	1	$P_u(B) = P(B)$	0.98
f	F	2	$P_f(B) = P(B\|F)$	0.92
d	D	2	$P_d(B) = P(B\|D)$	0.67
fd	F,D	4	$P_{fd}(B) = P(B\|F, D)$	0.64

2.2. Source separation based on contextual features

Given a feature space, defined on the unary bins of the UBC, each feature describes a contextual situation of the bin associated to it. These contextual situations, simply called situations hereafter, are then used to classify bins into categories with similar statistical behavior.

Let us consider coding of the significance bin, which indicates whether a coefficient is non-zero or zero. A 2D feature space is defined for the significance bin by 1) its frequency band, and 2) its neighborhood density.

To evaluate the impact of the significance bin source separation with the above features, a dataset of coded significance bins, from 4×4 blocks was prepared. In this experiment, a frequency band is considered as "low" if it is located in the top-left part of the block (otherwise, "high"). Moreover, the neighborhood density of a coefficient is considered as a "low" density, if less than half of its available neighbors are significant (otherwise, "high"). Now let B, F and D be three binary random variables corresponding to the significance bin, the frequency band and the neighborhood density, respectively. By using either F or D for source separation of the significance bin B, four configurations (i.e. cfg) can be defined. Table 1 shows the efficiency of each configuration in terms of the significance bin entropy H, calculated as:

$$H_{\text{cfg}} = -\sum_{b=0,1} P_{\text{cfg}}(B = b) \times \log_2 P_{\text{cfg}}(B = b), \quad (1)$$

As can be seen from this simple example, a source separation by using proper features can reduce the transmission rate of bins. In the proposed transform amplitude coding, we aim at using the bitplane representation of transform blocks to provide a simple contextual information access for feature extraction. A classifier is then trained on the extracted features to optimize the source separation scheme constrained by the final number of clusters.

2.3. Situation reduction by K-Means

The number of situations can easily become too large for being coded by exclusive CABAC context models. To address this, the second step of the proposed framework applies a K-Means algorithm to categorize situations based on their statistical behavior.

Let D be a dataset of actual samples from coded video streams with different QPs. Each entry of this dataset is the unary bitplane representation of one unsigned amplitude block of size 4×4, as shown in Fig. 2. Assuming that D contains a total of N_{bin} bins, one can define a feature space and extract feature vectors from all bins. Given the fact that the feature vectors in this space represent situations, a space with N_{sit} situations actually separates the significance bin information source into N_{sit} sub-sources. The main goal of K-Means is to derive a coarser source separation scheme with N_{ctx} coding contexts. The output of this process is supposedly a table T to map each of N_{sit} situations into one of N_{ctx} coding contexts. Algorithm 1 describes the proposed K-Means algorithm.

Algorithm 1 K-Means algorithm for reduction of N_{sit} situations into N_{ctx} coding contexts.

```
1: procedure K-MEANS
2:     Initialize randomly entries of T with 1, ..., N_ctx.
3:     Initialize: sit ← 1
4: top:
5:     if sit > N_sit then
6:         return T
7:     Initialize: ctx ← 1,  ctx* ← −1,  Rate* ← ∞
8: loop:
9:     if ctx > N_ctx then
10:        T[sit] ← ctx* // assign the best context
11:        sit ← sit + 1
12:        goto top
13:    else
14:        T[sit] ← ctx
15:        Rate ← getRate(T)
16:        if Rate < Rate* then
17:            Rate* ← Rate
18:            ctx* ← ctx
19:        ctx ← ctx + 1
20:    goto loop
```

As can be seen in Algorithm 1, the context of each situation sit is assigned after minimizing a rate function. This greedy approach uses the latest context assignment stored in the table T and attempts to update $T[sit]$ with a new context ctx^* with lowest rate, if possible.

In order to compute the rate of the signal in D, given the situation to context table T, the algorithm uses an entropy-based rate of the signal. Let B be the binary random variable of bins in D and $G(B)$ a function that extracts features of B and returns its situation index. Also define D_i as the subset of D containing all the bins whose situations are mapped to context ctx_i, as:

$$D_i = \{B | B \in D, T[G(B)] = ctx_i\}. \tag{2}$$

The probability distribution function of the sub-source separated by context ctx_i can be expressed as:

$$P_i(B) = P(B | B \in D_i). \tag{3}$$

Moreover, its sub-source entropy is computed as:

$$Rate_i = -\text{Len}(D_i) \sum_{b=0,1} \log_2 P_i(B = b), \tag{4}$$

where $\text{Len}(D_i)$ is the number of bins in D_i, such as:

$$\sum_{i=0}^{N_{\text{ctx}}-1} \text{Len}(D_i) = N_{\text{bin}}. \tag{5}$$

Therefore, the total rate of the dataset D is calculated as:

$$Rate = \sum_{i=0}^{N_{\text{ctx}}-1} Rate_i = -N_{\text{bin}} \sum_{i=0}^{N_{\text{ctx}}-1} \sum_{b=0,1} \log_2 P_i(B = b). \tag{6}$$

Experiments show that adequate iterations with the above K-Means algorithm can guarantee a convergence.

3. PROPOSED UBC CONFIGURATION

In this section, the proposed configuration of the UBC framework is explained. This algorithm replaces the coefficient coding of VVC for 4×4 blocks to address its inefficiency.

3.1. Feature space definition

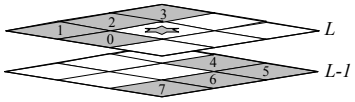
Three bin level features are used in the proposed UBC configuration. These features are namely the neighborhood density v_d, the bitplane number v_l and the frequency band v_f and compose a feature vector $V =< v_d, v_l, v_f >$ for each bin.

Neighborhood density v_d: To model the spatial density at the level L of a coefficient, we define a small neighborhood of 3×3 around it, as depicted in Fig. 3. As can be seen, the current bin to encode (starred) has access to its causal neighbors at level L and its non-causal neighbors at level $L - 1$. With a maximum of eight available neighbors, $nb_i,\ i = 0, 1, ..., 7$, as shown with gray cells in Fig. 3, the density feature can have $2^8 = 256$ values and is calculated as:

$$v_d = \sum_{i=0}^{7} 2^{nb_i}. \tag{7}$$

Bitplane number v_l: The second proposed feature to extract from a bin is the bitplane level L it is located in. Technically, the upper bound of coefficient amplitudes is 65635. However, very large amplitudes rarely occur with conventional

Fig. 3. The 3×3 neighborhood around a coefficient at levels L and $L-1$.



QPs. Therefore, to limit the value of v_l, a threshold at level 15 is applied to give 16 different v_l values:

$$v_l = min(L, 15). \tag{8}$$

Frequency v_f: There are 16 different frequencies inside a 4×4 transform block. The frequency feature value in the proposed UBC algorithm uses the scan order index.

3.2. Bucketizing the feature space

In order to apply the proposed UBC algorithm, the look-up table T, explained in the previous section, needs to be stored at both encoder and decoder sides. As explained, each entry of T corresponds to one situation defined by a feature vector $V =< v_d, v_l, v_f >$ and contains the CABAC context model number associated to that situation. Therefore, with no feature space reduction, this table should have an excessive number of $256 \times 16 \times 16 = 65536$ entries ($v_d \times v_l \times v_f$), which is expensive to store, especially at the decoder side.

To address this problem, the feature space is bucketized with another round of K-Means to quantize the domain of the features. In this round, an offline brute-force search is carried out on each feature separately, in order to find the best bucketizing scheme. During the search process of each feature, other features are used with their original domain to evaluate different bucketizing schemes on the current feature. The result of each search is a bucket table to map the original domain of its corresponding feature to a bucketized domain. Finally, all three bucket tables are integrated to be used before the situation to context table *T*, which now has a smaller size, due to the smaller domain of each feature.

The outcome of the bucketizing step is a coarser representation of the feature space of *V* which would cause a performance drop compared to the full domain feature space. Therefore, a compromise has been made between the performance and the bucket tables sizes. As a result, three bucket tables have been optimized to reduce the original feature space size of 65536 to $30 \times 4 \times 4 = 480$ situations ($v_d \times v_l \times v_f$).

4. RESULTS

For performance evaluation, the transform coding of JEM5 is replaced by the UBC for all 4×4 blocks [17]. This is performed by storing a situation to context table *T* of size 480, as well as three bucket tables at both encoder and decoder sides. Table 2 compares the coding performance of UBC against JEM on the screen content of the JVET and the HEVC-SCC Common Test Conditions (CTCs) [18, 19]. In order to avoid over-training of *T*, none of the test sequences in Table 2 was used. As can be seen, the UBC algorithm improves BD-R [20] performance of JEM by 2.8% and 3.4% in the random access and all intra modes, respectively. The better performance in the all intra mode could be due to the higher density of 4×4 blocks in intra slices than inter slices as shown in Table 3.

Table 2. Performance of the UBC against JEM, in terms of BD-rate gain (%) and coding time (%).

Res.	Sequence	Random access		All intra	
		BD-rate	ET/DT	BD-rate	ET/DT
2560	Basketball_Sc	-2.74	125/114	-2.70	133/116
×	MissionCtrlClip2	-2.82	147/120	-2.99	150/128
1440	**Average**	**-2.78**	**136/117**	**-2.84**	**141/122**
	FlyingGraphics	-1.55	130/125	-2.26	131/128
	Desktop	-4.42	138/126	-6.57	146/135
	Console	-3.05	140/132	-6.48	146/140
1920	ChineseEditing	-3.23	140/129	-3.16	147/137
×	MissionCtrlClip3	-2.68	125/134	-3.31	148/137
1080	Robot	+0.04	133/116	+0.06	132/124
	ChinaSpeed	-1.38	107/118	-3.58	138/129
	Average	**-2.32**	**128/126**	**-3.68**	**141/133**
	Web_browsing	-4.88	132/120	-4.10	136/122
1280	Map	-0.98	138/109	-1.06	147/121
×	Programming	-4.51	135/104	-4.88	139/112
720	SlideShow	-3.06	140/119	-3.42	149/130
	SlideEditing	-3.93	126/108	-5.19	134/118
	Average	**-3.47**	**115/112**	**-2.84**	**147/121**
	Total Average	**-2.8**	**126/119**	**-3.4**	**141/127**

5. CONCLUSION

A transform coefficient coding algorithm, Unary Bitplane Coding, is proposed to improve the decorrelation remaining in the quantized coefficients of screen content. For this purpose, a set of contextual situations are defined on unary bitplanes of coefficients and then are merged to a few coding contexts by CABAC. The experiments show that UBC achieves an average gain of 2.8% and 3.4% in the random access and all intra modes, respectively.

Table 3. Statistics of 4×4 blocks in intra and inter slices of screen contents.

QP	Population		Rate	
	Inter	Intra	Inter	Intra
22	53%	51%	19%	36%
27	52%	52%	15%	37%
32	43%	56%	10%	38%
37	3%	55%	1%	37%

6. REFERENCES

[1] M. Wien, V. Baroncini, J. Boyce, A. Segall, and T. Suzuki. Preliminary joint call for evidence on video compression with capability beyond HEVC. In *Document JVET-E1002 5th JVET Meeting, Geneva, Switzerland*, January, 2017.

[2] X. Xu, S. Liu, T.D. Chuang, Y.W. Huang, S.M. Lei, K. Rapaka, C. Pang, V. Seregin, Y.K. Wang, and M. Karczewicz. Intra block copy in HEVC screen content coding extensions. *IEEE Journal on Emerging and Selected Topics in Circuits and Systems*, 6(4):409–419, 2016.

[3] L. Guo, J. Sole, and M. Karczewicz. Palette mode for screen content coding, document JCTVC-M0323, Incheon. *Korea*, April, 2013.

[4] P. Lai, S. Liu, and S. Lei. AHG6: On adaptive color transform (ACT) in SCM2. 0. *19th Meeting Joint Collaborative Team Video Coding JCTVC-S0100, Strasbourg, France*, October, 2014.

[5] L. Zhang, X. Xiu, J. Chen, M. Karczewicz, Y. He, Y. Ye, J. Xu, J. Sole, and W.S. Kim. Adaptive color-space transform in HEVC screen content coding. *IEEE Journal on Emerging and Selected Topics in Circuits and Systems*, 6(4):446–459, 2016.

[6] B. Li, J. Xu, G.J. Sullivan, Y. Zhou, and B. Lin. Adaptive motion vector resolution for screen content. In *Proc. 19th Meeting Joint Collaborative Team Video Coding-S0085, Strasbourg, France*, pages 1–14, October, 2014.

[7] M. Abdoli, G. Clare, F. Henry, and P. Philippe. AHG11: Block DPCM for screen content coding. In *Document JVET-L0078, Macau, China*, October, 2018.

[8] J. Xu, R. Joshi, and R.A. Cohen. Overview of the emerging HEVC screen content coding extension. *IEEE Transactions on Circuits and Systems for Video Technology*, 26(1):50–62, 2016.

[9] J. Sole, R. Joshi, N. Nguyen, T. Ji, M. Karczewicz, G. Clare, F. Henry, and A. Duenas. Transform coefficient coding in HEVC. *IEEE Transactions on Circuits and Systems for Video Technology*, 22(12):1765–1777, 2012.

[10] M. Mrak and J. Xu. Improving screen content coding in HEVC by transform skipping. In *2012 Proceedings of the 20th European Signal Processing Conference (EUSIPCO)*, pages 1209–1213, Aug 2012.

[11] Y. H. Tan, C. Yeo, and Z. Li. Residual DPCM for lossless coding in HEVC. In *2013 IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 2021–2025, May 2013.

[12] M. Naccari, M. Mrak, A. Gabriellini, S. Blasi, and E. Izquierdo. Inter prediction residual DPCM, document JCTVC-M0442, Incheon. *Korea*, April, 2013.

[13] R. Joshi, J. Sole, and M. Karczewicz. AHG8: Residual DPCM for visually lossless coding, document JCTVC-M0351, Incheon. *Korea*, April, 2013.

[14] M. Abdoli, F. Henry, P. Brault, P. Duhamel, and F. Dufaux. Intra prediction using in-loop residual coding for the post-HEVC standard. In *IEEE 19th International Workshop on Multimedia Signal Processing (MMSP), Luton, United Kingdom*, pages 1–6, October 2017.

[15] M. Abdoli, F. Henry, P. Brault, P. Duhamel, and F. Dufaux. Short-distance intra prediction of screen content in versatile video coding (VVC). *IEEE Signal Processing Letters*, 25(11):1690–1694, Nov 2018.

[16] J. Han, C. Chiang, and Y. Xu. A level-map approach to transform coefficient coding. In *Image Processing (ICIP), 2017 IEEE International Conference on*, pages 3245–3249, 2017.

[17] J. Chen, E. Alshina, G. Sullivan, J. Ohm, and J. Boyce. Algorithm description of joint exploration test model 5 (JEM 5). In *Document JVET-E1001, Geneva, Switzerland*, January, 2017.

[18] K. Suehring and X. Li. JVET common test conditions and software reference configurations. In *Document JVET-H1010 8th JVET Meeting, Macau, China*, October, 2017.

[19] H. Yu, R.A. Cohen, K. Rapaka, and J. Xu. Common test conditions for screen content coding. In *ITU-T SG 16 WP 3 and ISO/IEC JTC 1/SC 29/WG 11, 26th Meeting: Geneva, CH, 12 – 20*, 2015.

[20] G. Bjontegaard. Calculation of average PSNR differences between rd-curves. In *ITU-T Q. 6/SG16 VCEG, 15th Meeting, Austin, Texas, USA*, April, 2001.