STRUCTURED "TRUNCATED GOLOMB CODE" FOR CONTEXT-BASED ADAPTIVE VLC

Sadaatsu KATO, Kazuo SUGIMOTO, Satoru ADACHI, and Minoru ETOH

Multimedia Laboratories, NTT DoCoMo, Inc.
3-5, Hikarinooka, Yokosuka, Kanagawa, Japan

ABSTRACT

In this paper, we describe a structured variable length coding (VLC) based on Golomb code that generalize the current context-based adaptive variable length coding (CAVLC) in the emerging video coding standard H.264/AVC. The current CAVLC is characterized as unstructured VLC and uses extensive dedicated code tables. Lack of generality due to these code tables causes a problem of over-fitting or over-learning when we estimate a set of coding parameters. We propose a "compact" structured code based on Golomb code providing generality, extensibility, and low implementation complexity with only three parameters. Introduction of a simple truncation method enables to avoid reversed order code length and remove unused code space for finite set of symbols. Experimental results show its similar coding efficiency to current CAVLC while reducing the size of memories to store code tables and providing adaptability for various probability functions.

1. INTRODUCTION

Context-based adaptive variable length coding (CAVLC) [1] has been adapted in the emerging video coding standard H.264/AVC [2], and it provides considerable improvement of coding efficiency over the conventional DCT coefficient coding by UVLC [3]. The current CAVLC is characterized as unstructured VLC and uses extensive dedicated code tables. Therefore, lack of generality and large size of memory requirement for these tables are still remained to be solved.

In this paper, we propose a method to generalize the dedicated unstructured VLC in the current CAVLC with structured VLC based on Golomb code [4]. We introduce a simple "truncation" rule to provide compactness, which has no reversed order in code length and no redundant code space for finite set of symbols. We call the structured VLC "Truncated Golomb code." Keeping the current CAVLC framework intact, the introduction of this code has following advantages.

1. Generality of code table design

If we design a VLC table specifically tuned for several test sequences, the dedicated VLC table always outperforms over a generic VLC for the test sequences. When encoding other sequences, however, optimality of the dedicated code table is not guaranteed. In general, we have been suffering from over-fitting or over-learning problem in parameter estimation. By increasing the number of model parameters, the expected parameter estimation error is increased in such design as discussed in an extensive literature. If we can model the probability density function (PDF) with fewer parameters and the performance is similar to the dedicated VLC tables, the parametric VLC table has performance stability in general sequences.

2. Extensibility toward future improvement

Truncated Golomb code provides structured VLC tables parameterized by three variables: one for the number of symbols and two for PDF representation. Parametric description is very important for future evolution of H.264/AVC codec, since we will be able to avoid introducing dedicated VLC tables. Conditional PDF is also easily modeled by associating PDF parameters with the context.

3. Low implementation complexity

By the definition of Golomb code, VLC decoders can be realized by straightforward table matching. No state-transition decoders (automata) are required. Moreover, if we need more code tables for future improvement, truncated Golomb code provides implementation commonality for each VLC decoder, while an additional dedicated VLC requires us to implement another VLC decoder.

According to the abovementioned advantages, adaptation of truncated Golomb code to the current CAVLC is promising if its performance is considerably similar to that of the current dedicated VLC. In the following sections, we will describe coding design and experimental results.

2. A "COMPACT" STRUCTURED CODE: TRUNCATED GOLOMB CODE

2.1 Problem to be solved

Structured VLC such as Golomb code does not need to store its table since its correspondence between symbols and codes can be mathematically defined. Change of its parameter can provide various PDF without increase of memory for the table.

However, structured VLC is not appropriate to code small number of symbols such as the Run tables of CAVLC. Table 1 shows an example. (a) shows a Golomb code table of p=2 and q=0. Here, p denotes that the binary part provides p variations to a unary code, and q denotes that the binary part is appended to a code after code number q (q=0 means all of the codes have the binary part). If an element to be coded has limited number of symbols, the Golomb code table is truncated. But simple truncation results in unused code space, thus it decreases coding efficiency (Table 1 (b)). To remove unused code space, codes which have unused code space can be changed. But it results in reversed order of code length (Table 1 (c)) and it decreases coding efficiency anyway.

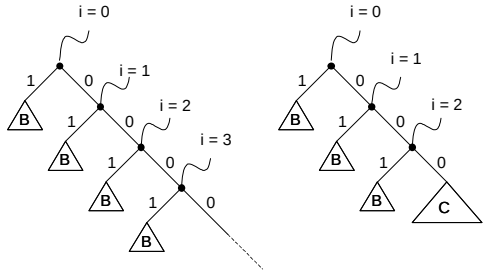
Table 1. A Golomb code table and its truncation into a 7 symbol table. (*p=2, q=0*)

Code number	(a)	(b)	(c)
0	1 0	1 0	1 0
1	1 1	1 1	1 1
2	01 0	01 0	01 0
3	01 1	01 1	01 1
4	001 0	001 0	001 0
5	001 1	001 1	001 1
6	0001 0	0001 0	000
7	0001 1		
8	00001 0		
...	...		

To cope with this problem that prevents introduction of structured VLC into CAVLC, we introduce a simple truncation mehod.

2.2 Truncation method

Figure 1 illustrates a Golomb code tree explaining the basic truncation rule. Figure 1 (a) shows a code tree of a Golomb code table. Sub tree B corresponds to the binary part of Golomb code. Figure 1 (b) shows truncated one. Here, the tree is truncated at a node i=3 or at a node within sub tree B to limit the number of symbols. In this case, the tree may have reversed order of code length. We truncate the tree at the upper node, and attach a new sub tree C that has appropriate number of nodes to limit the number of total symbols.



(a) A code tree of a Golomb code (b) A truncated tree with a new sub tree

Figure 1. Illustration of the truncation rule on a code tree.

Table 2 shows Golomb code tables. These tables serve as a base table to be truncated and provide various PDF with their parameter (p, q). Table 3 shows small tables to be required as a sub table that corresponds to the sub tree C attached to a node of a Golomb code tree in Figure 1. Here, we have tables of 2, 3, 4, 5, 6, 7, and 8 symbols, and their variation of PDF properties. Those tables do not have reversed order of code length nor redundant code space. Therefore, if the length of a shortest code from a sub table C is equal or longer than that of a longest code from a sub tree B, we can obtain a table of limited number of symbols, which do not have reversed order of code length nor redundant code space.

Table 2. Golomb tables as base tables.

Code No.	P=2, q=0	p=2, q=1	p=2, q=0	p=3, q=0	P=4, q=0
0	1	1	1 0	1 0	1 00
1	01	01 0	1 1	1 10	1 01
2	001 0	01 1	01 0	1 11	1 10
3	001 1	001 0	01 1	01 0	1 11
4	0001 0	001 1	001 0	01 10	01 00
5	0001 1	0001 0	001 1	01 11	01 01
6	00001 0	0001 1	0001 0	001 0	01 10
7	00001 1	00001 0	0001 1	001 10	01 11
8	000001 0	00001 1	00001 0	001 11	001 00
9	000001 1	000001 0	00001 1	0001 0	001 01
10	0000001 0	000001 1	000001 0	0001 10	001 10
11	0000001 1	0000001 0	000001 1	0001 11	001 11
12	00000001 0	0000001 1	0000001 0	00001 0	0001 00
13	00000001 1	00000001 0	0000001 1	00001 10	0001 01
14	000000001 0	00000001 1	00000001 0	00001 11	0001 10
...	...	...	...	...	...

Table 3. Small tables as sub tables.

Code No.	Table 2	Table 3	Table 4-1	Table 4-2	Table 5	Table 6-1	Table 6-2	Table 7	Table 8
0	1	1	1	11	11	11	11	11	111
1	0	01	01	10	10	10	10	101	110
2		00	001	01	01	01	011	100	101
3			000	00	001	001	010	011	100
4					000	0001	001	010	011
5						0000	000	001	010
6								000	001
7									000

Each table is assigned to Golomb code parameters p and q as shown in Table 4. L in Table 4 denotes "residual number" which means the number of nodes to be attached as a sub tree C in Figure 1 (b). The total number of symbols n and Golomb code parameters p and q gives L, as described below. To provide a table with flat PDF, we use simple binary code table in addition to the Golomb code based VLC.

Table 4. Combination of the tables relative to parameters of pdf.

Golomb code table	Sub table						
	L=2	L=3	L=4	L=5	L=6	L=7	L=8
p=2, q=2	Table 2	Table 3	Table 4-1				
p=2, q=1	Table 2	Table 3	Table 4-1				
p=2, q=0	Table 2	Table 3	Table 4-1				
p=3, q=0	Table 2	Table 3	Table 4-2	Table 5	Table 6-1		
p=4, q=0	Table 2	Table 3	Table 4-2	Table 5	Table 6-2	Table 7	Table 8
(binary)							

Combination of the tables is uniformly given from Golomb code parameters p, q, and a total number of symbols n. Thus encoding and decoding of a certain symbol/code can be mathematically defined.

Encoding process of an *m*-th index in a table (*p*, *q*, *n*);

1. Get *h*=(*n*-*q*)%*p*. in case of *h*==0, *h*=*p*.
2. Get "residual number" *L*=*h*+*p*. in case of *L*>*n*, *L*=*n*.
3. if *L*==*n*, a code is a code from the assigned sub table of *L* symbols. Output *m*-th code of the sub table.
4. if *m*<=*n*-*L*, a code is a Golomb code. Output *m*-th code of the assigned Golomb code table.
5. if *m*>*n*-*L*, a code is a truncated Golomb code. First, output "0"s (*n*-*L*+*q*)/*p* times as a preceding part of a code. Second, output a (*m*+*L*-*n*)-th code of the sub table of *L* symbols as a suffix of a code.

Decoding process of a code in a table (*p*, *q*, *n*);

1. Get *h*=(*n*-*q*)%*p*. in case of *h*==0, *h*=*p*.
2. Get "residual number" *L*=*h*+*p*. if *L*>*n*, *L*=*n*.
3. if *L*==*n*, a code is a code from the assigned sub table of *L* symbols. Output a corresponding index from the sub table.
4. Check the number of "0"s at the top of a code, *t*.
5. if *t*<(*n*-*L*+*q*)/*p*, a code is a Golomb code. Output a corresponding index from the assigned Golomb code table.
6. if *t*>=(*n*-*L*+*q*)/*p*, a code is a truncated Golomb code. From the bits after (*n*-*L*+*q*)/*p* "0"s, get a corresponding intermediate index *x* from the sub table of *L* symbols. Output an index *x*+*n*-*L*.

Using the above encoding/decoding rules, various tables with various PDF properties can be provided without additional memory to store tables.

3. ADAPTATION OF TRUNCATED GOLOMB CODE INTO CURRENT CAVLC ELEMENTS

Adaptation to the current NumCoef/Trailing1s, TotalZeros, and Run tables is described below. We do not replace Level tables of the current CAVLC since they are already Golomb based structured VLC.

3.1 NumCoef/Trailing1s

Since the original NumCoef/Trailing1s tables are 2D style VLC tables, we define a simple rule to assign a code number of the structured table to NumCoef/Trailing1s, such as the one used for the former UVLC coding of Level/Run combination.

Table 5 shows an example mapping of code numbers on NumCoef/Trailing1s table. For the NumCoef/Trailing1s combination there is a simple rule. The NumCoef/Trailing1s combinations are assigned a code number according to the priority: 1) Trailing1s 2) NumCoef (ascending).

FLC representation of NumCoef/Trailing1s, which can be chosen in addition to the VLC tables, is not modified. Table 6 shows the parameter of those tables.

3.2 TotalZeros

Some of the original TotalZeros in current CAVLC tables have its shortest code in the middle of the table. Therefore we define a "center" value for each table of TotalZeros as shown in Table 7, and assign a code number starting from the "center" as shown in Table 8. Table 9 and Table 10 show the parameter of those tables.

Table 5. Mapping of a code number on NumCoef/Trailing1s tables.

Trailing1s / NumCoef	0	1	2	3
0	0			
1	4	1		
2	8	5	2	
3	12	9	6	3
4	16	13	10	7
5	20	17	14	11
6	24	21	18	15
7	28	25	22	19
8	32	29	26	23
9	36	33	30	27
10	40	37	34	31
11	44	41	38	35
12	48	45	42	39
13	52	49	46	43
14	56	53	50	47
15	59	57	54	51
16	61	60	58	55

Table 6. Parameters for NumCoef/Trailing1s tables.

NumCoef/Trailing1s	p	q	n
Num-VLC0	2	2	62
Num-VLC1	4	0	62
Num-VLC2	binary		62
Chroma_DC	2	2	14

Table 7. "center" values for TotalZeros tables for all 4x4 blocks.

NumCoef	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Center	0	0	7	5	9	5	6	7	6	5	4	3	3	2	0

Table 8.Mapping of a code number on TotalZeros (TotalZeros table for Numcoef = 7).

Totalzeros	0	1	2	3	4	5	6	7	8	9
Mapping order	9	8	7	5	3	1	0	2	4	6

Table 9. Parameters for TotalZeros tables for all 4x4 blocks.

NumCoef	p	q	n
1	3	0	16
2	4	0	15
3	4	0	14
4	4	0	13
5	4	0	12
6	3	0	11
7	2	0	10
8	2	0	9

NumCoef	p	q	n
9	2	0	8
10	2	0	7
11	3	0	6
12	2	1	5
13	2	0	4
14	2	0	3
15	2	0	2

Table 10. Parameters for TotalZeros tables for all 4x4 blocks for chroma DC 2x2 blocks.

NumCoef	p	q	n
1	2	0	4
2	2	0	3
3	2	0	2

3.3 Run

The proposed tables can immediately replace the original

Run tables. We simply define *p*, *q*, and *n* to each table. Table 11 shows those parameters and actual codes.

Table11. Parameters for Run tables.

Run left	p	q	n
1	2	0	2
2	2	0	3
3	2	0	4
4	2	0	5
5	2	0	6
6	3	0	7
> 6	2	0	15

4. EXPERIMENTS

4.1 Simulation Conditions

We adapt the new truncated Golomb code into current H.26L and compare its coding efficiency with the original unstructured CAVLC. Our coder is based on the H.26L test model JM1.1 [5] and each CAVLC table is replaced to the proposed truncated Golomb code created by three parameters. Test sequences are Container, Foreman, News (QCIF, 10fps), Silent (QCIF, 15fps), Mobile (CIF 15fps), Paris, and Tempete (CIF, 30fps). Motion search resolution is 1/4 pel and the number of multiple reference frames is set to 5. BDbitrate values [6] of low QP range (0, 4, 8, and 12, in JM1.1) and high QP (16, 20, 24, and 28 in JM1.1) are calculated.

4.2 Parameter setting for whole conditions

Table 12 shows BDbitrates of truncated Golomb code relative to the unstructured VLC of the current CAVLC. Positive value of BDbitrate denotes increase of bitrates and negative values denotes decrease of bitrates from the unstructured VLC. Truncated Golomb code provides similar coding efficiency (on average 1% in BD bitrate) in both Low QP range and High QP range.

Table 12. BDbitrates of truncated Golomb code relative to the unstructured CAVLC for whole conditions.

Sequence	BD bitrate [%]			
	Intra		Inter	
	Low QP	High QP	Low QP	High QP
Container	0.28	1.50	1.52	0.71
Foreman	0.24	1.30	1.23	1.58
News	-0.60	0.80	0.80	0.88
Silent	-0.75	0.12	1.52	1.83
Mobile	-0.49	0.73	0.89	1.26
Paris	-0.58	0.66	0.76	1.66
Tempete	-0.67	0.27	1.00	1.07
Average	-0.34	0.76	1.10	1.28

4.3 Parameter setting for Intra/Inter pictures and QPs

Since the truncated Golomb code provides various tables without increase of VLC tables, additional table selection can be introduced to achieve more accurate modeling of the PDF. Table 13 shows BDbitrates of truncated Golomb code with distinct parameters (*p*, *q*) for each picture mode (Intra/Inter) and QP. The results show that additional table and its selection can increase coding efficiency of CAVLC, and the parametric description of the truncated Golomb code can easily achieve it.

Table 13. BDbitrates of truncated Golomb code relative to the unstructured CAVLC with parameter setting for each condition.

Sequence	BD bitrate [%]			
	Intra		Inter	
	Low QP	High QP	Low QP	High QP
Container	0.28	1.17	0.94	0.46
Foreman	0.26	0.82	1.12	0.55
News	-0.60	0.69	1.40	0.46
Silent	-0.72	-0.07	1.86	1.35
Mobile	-0.58	0.32	0.45	0.47
Paris	-0.63	0.63	0.64	0.54
Tempete	-0.64	-0.20	0.63	0.08
Average	-0.38	0.48	1.00	0.56

4.4 Comparison of memory requirement

The original NumCoef/Trailing1s, TotalZeros, and Run tables of unstructured CAVLC have 29 distinct tables (4 NumCoeff/ Trailing1s tables, 18 TotalZeros tables, and 7 Run tables) and more than 400 entries of codes.

On the other hand, using the above-mentioned truncated Golomb code, each table can be specified with parameters *p*, *q*, and *n* (and "center" for NumCoef/Trailing1s tables). The sub tables must be stored, but they have only 45 entries of codes. Therefore, truncated Golomb code tables require significantly less memory to store them.

Moreover, it should be noted that parametric representation of the tables provides adaptability for future evolution by simply assigning new value of three parameters. No additional code tables are required for this purpose.

5. CONCLUSIONS

We proposed *Truncated Golomb code* to replace the unstructured VLC in the current CAVLC. The compact structured VLC offers generality, extensibility, and low complexity into CAVLC and simulation results shows its similar coding efficiency in comparison with the dedicated unstructured VLC while reducing the size of memories and providing adaptability for future evolution.

Further studies include automatic parameter choice according to the context with learning process and introduction of other structured code, such as Exp-Golomb code, to provide more variety of pdf.

REFERENCES

[1] G. Bjøntegaard and Karl Lillevold, "Context-adaptive VLC (CVLC) coding of coefficients", document JVT-C028, JVT of ISO/IEC MPEG & ITU-T VCEG, 3rd Meeting, Fairfax, Virginia, USA, 6-10 May, 2002

[2] T. Wiegand, "Joint Final Committee Draft (JFCD) of Joint Video Specification (ITU-T Rec. H.264 | ISO/IEC 14496-10 AVC)", document JVT-D157, JVT of ISO/IEC MPEG & ITU-T VCEG, 4th Meeting, Klargenfurt, Austria, 22-26 July, 2002

[3] Y.Itoh, and N.-M. Cheung, "Universal variable length code for DCT coding", Proc. IEEE int. Conf. Image Processing (ICIP), Vancouver, Canada, Sept. 10-13, 2000

[4] S.W.Golomb, "Run-length encoding", IEEE trans.IT-12, pp399-401, 1966

[5] ftp://standard.pictel.com/video-site/h26L/jm11.zip

[6] G.Bjontegaard, "Calculation of average PSNR differences between RD-curves", VCEG-M33, Apr, 2001