INTERNATIONAL ORGANISATION FOR STANDARDISATION
ORGANISATION INTERNATIONALE DE NORMALISATION
ISO/IEC JTC1/SC29/WG11
CODING OF MOVING PICTURES AND AUDIO

ISO/IEC JCTC1/SC29/WG11 MPEG/m53678
April 2020, Online

Source:	Apple Inc.	
Status:	Input document	
Title:	G-PCC: Coding of significant attribute coefficients	
Author(s):	David Flynn	*davidflynn@apple.com*
	Khaled Mammou	*kmammou@apple.com*

Abstract

Attribute coding in the current G-PCC draft represents each coefficient tuple by a significant coefficient tuple flag. If the coefficient tuple is not significant, further syntax elements relating to the coefficient values are skipped. While a run-length coding process is described, the use of a unary run-length representation results in numerous bins. This contribution proposes a more efficient representation.

Run-length coding

Attribute coding uses zero-run-length coding to encode significant coefficient flags. The current binarisation of the zero-run is truncated unary coded and results in as many bins as there are coefficients.

To illustrate, the following sequence of significant coefficient flags are converted to run lengths and then coded using a unary code:

```
sig_coeff:   0 1 0 0 1 1 0 0 0 1 1 1
run_length: 1   2 . . 0 3 . . . 0 0
unary:       0 1|0 0 1|1|0 0 0 1|1|1
```

As a brief aside, the current use of truncated unary coding instead of unary coding makes little sense where the truncation can occur at most once per slice. To this end, it would be better to use unary coding and avoid the extra truncation check.

In the current design, contextualisation is performed using a sequence of three contexts for each zero run.

Proposal

While a unary code is suitable for coding distributions with an exponential distribution, a practical design can switch to a code that grows at a slower rate after coding an initial interval. Such codes are sub-exponential codes, or a concatenation of a Golomb-Rice code with an exponential Golomb code.

However, replacing the coding of zero-runs only addresses half of the problem. While long zero runs are expected with high degrees of quantisation, as quantisation decreases, or in the limit at lossless coding, the opposite occurs: long runs of ones.

To enable a reduction of coded bins and context coded bins, it is proposed to use a two state run-length coder that codes zero-runs and non-zero-runs.

The system operates as follows:

A decoder tracks the current run state (zero-run or non-zero-run). Each coded run-length indicates the termination of the current run. Therefore the end of a run indicates the inversion of the run state. The initial run state is explicitly signalled at the start of a slice. Each run is binarised using a k-th order sub-exponential code, with the unary prefix being coded with up to five contexts. Contextualisation is performed independently for each type of run.

Since the system switches between the two states, it is impossible to signal a run-length of 0 (which would indicate a return to the current state). Each run-length is coded as run_length_minus1.

To illustrate using the same example:

```
sig_coeff:           0 1 0 0 1 1 0 0 0 1 1 1
run:                1 1 2 . 2 . 3 . . 3 . .
init_run_type:      0
run_length_minus1:  0 0 1 . 1 . 2 . . 2 . .
```

The following summarises the decoding process:

```
// NB: initial runType is inverted to make loop more readable
runType = 1 ^ decodeRunType();
runLength = 0
for (idx = 0; idx < numPoints; idx++) {
  if (!runLength--) {
    // end of run, decode next run length
    // run type inverts with each run
    runLength = decodeRunLengthMinus1(runType);
    runType = !runType;

    // If starting a run of zeros, process here and jump to next run
    if (runType == 0) {
      fill(coeff[i], runLength + 1, 0);
      idx += runLength;
      runLength = 0;
      continue;
    }
  }

  // NB: coefficient is known not to equal 0 since this isn't a run of zeros.
  coeff[idx] = decodeCoeff();
}
```

The performance of the proposed method is assessed according to the common test conditions [1] and compared to the TMC13v9.0 anchor [2] in Tables 1 and 2.

Table 1 – Performance of proposed significant coefficient run-length coding using LoD attribute coding compared to v9.0 anchor

		BPP Ratio [%]			BD-Rate [Δ%]						Avg. of ratio maxrssk [%]		Ratio of avg. runtime [%]	
Condition	Class	Geometry	Colour	Refl	D1	D2	Y	Cb	Cr	R	Encoder	Decoder	Encoder	Decoder
C1_ai	cat1-A				0.0	0.0	−2.5	−4.5	−5.1		101	100	99	99
C1_ai	cat1-B				0.0	0.0	−0.3	−1.2	−1.8		100	100	99	97
C1_ai	cat3-fused				0.0	0.0	0.1	0.1	0.1	0.3	100	100	99	99
C1_ai	cat3-frame				0.0	0.0				−0.6	100	100	104	104
C1_ai	overall				0.0	0.0	−1.3	−2.6	−3.2	−0.3	100	100	100	99
C2_ai	cat1-A				0.0	0.0	−0.4	−0.6	−0.7		100	100	101	94
C2_ai	cat1-B				0.0	0.0	−0.1	−0.6	−1.0		100	100	101	100
C2_ai	cat3-fused				0.0	0.0	0.2	0.2	0.2	0.6	101	100	98	99
C2_ai	cat3-frame				0.0	0.0				0.6	100	100	99	96
C2_ai	overall				0.0	0.0	−0.2	−0.6	−0.8	0.6	100	100	101	97
CW_ai	cat1-A	100.0	100.1								100	100	96	98
CW_ai	cat1-B	100.0	100.0								100	100	101	104
CW_ai	cat3-fused	100.0	100.0	100.2							100	100	98	81
CW_ai	cat3-frame	100.0		100.2							100	100	98	97
CW_ai	overall	100.0	100.1	100.2							100	100	99	100
CY_ai	cat1-A				0.0	0.0	0.7	0.7	0.7		100	100	101	97
CY_ai	cat1-B				0.0	0.0	0.6	0.6	0.6		100	100	99	103
CY_ai	cat3-fused				0.0	0.0	0.4	0.4	0.4	1.1	100	100	97	94
CY_ai	cat3-frame				0.0	0.0				0.5	100	100	102	100
CY_ai	overall				0.0	0.0	0.6	0.6	0.6	0.7	100	100	100	100

NOTE — Condition CY metrics reported using Hausdorff PSNR.

Table 2 – Performance of proposed significant coefficient run-length coding using RAHT attribute coding compared to v9.0 anchor

		BPP Ratio [%]			BD-Rate [Δ%]						Avg. of ratio maxrssk [%]		Ratio of avg. runtime [%]	
Condition	Class	Geometry	Colour	Refl	D1	D2	Y	Cb	Cr	R	Encoder	Decoder	Encoder	Decoder
C1_ai	cat1-A				0.0	0.0	−4.4	−5.4	−5.9		101	101	97	99
C1_ai	cat1-B				0.0	0.0	−1.2	−1.8	−2.0		100	100	98	99
C1_ai	cat3-fused				0.0	0.0	0.1	0.2	0.2	−0.1	100	100	102	99
C1_ai	cat3-frame				0.0	0.0				−1.0	100	100	100	100
C1_ai	overall				0.0	0.0	−2.6	−3.3	−3.6	−0.7	100	100	98	99
C2_ai	cat1-A				0.0	0.0	−1.2	−1.4	−1.3		100	101	102	95
C2_ai	cat1-B				0.0	0.0	−0.8	−1.3	−1.5		100	100	101	97
C2_ai	cat3-fused				0.0	0.0	0.2	0.2	0.3	0.4	100	100	98	96
C2_ai	cat3-frame				0.0	0.0				−0.1	100	100	100	100
C2_ai	overall				0.0	0.0	−0.9	−1.3	−1.3	0.1	100	100	101	97

Sign flags

The current attribute coding design encodes signed coefficient values, x by converting to $2|x| + \text{sign}$. The resulting coefficient is then coded using the dictionary coder. However, the sign bits are unpredictable and serve to decrease the useful occupancy of dictionary (each magnitude will be represented twice), dilute the probability of a given magnitude, and reduce the effectiveness of coding greater than x flags.

To address this, we propose coding coefficients using separate magnitude and sign values. The sign value is coded as a bypass bin.

The following illustrates the changes to the coefficient decoding:

```
@@ -159,18 +159,22 @@ PCCResidualsDecoder::decode(uint32_t value[3])
   int b3 = value[1] <= 1;
   value[2] = decodeSymbol(3 + (b0 << 1) + b2, 3 + (b1 << 1) + b3, 1);

-  int d = (value[0] == value[1] && value[0] == value[2]);
-  for (int k = 0; k < 3; k++) {
-    value[k] += d;
-  }
+  if (value[0] && arithmeticDecoder.decode(binaryModel0))
+    value[0] = -value[0];
+  if (value[1] && arithmeticDecoder.decode(binaryModel0))
+    value[1] = -value[1];
```

```
+  if (value[2] && arithmeticDecoder.decode(binaryModel0))
+    value[2] = -value[2];
 }

 //--------------------------------------------------------------------------

-uint32_t
+int32_t
 PCCResidualsDecoder::decode()
 {
-  return decodeSymbol(0, 0, 0) + 1;
+  auto mag = decodeSymbol(0, 0, 0) + 1;
+  bool sign = arithmeticDecoder.decode(binaryModel0);
+  return sign ? -mag : mag;
 }
```

The performance of the proposed method is assessed according to the common test conditions [1] and compared to the TMC13v9.0 anchor [2] in Tables 3 and 4.

Table 3 – Performance of proposed coefficient sign coding using LoD attribute coding compared to v9.0 anchor

		BPP Ratio [%]			BD-Rate [Δ%]						Avg. of ratio maxrssk [%]		Ratio of avg. runtime [%]	
Condition	Class	Geometry	Colour	Refl	D1	D2	Y	Cb	Cr	R	Encoder	Decoder	Encoder	Decoder
C1_ai	cat1-A				0.0	0.0	−0.7	−0.6	−0.6		100	100	97	99
C1_ai	cat1-B				0.0	0.0	−0.5	−0.5	−0.5		100	100	96	96
C1_ai	cat3-fused				0.0	0.0	−0.5	−0.6	−0.6	−0.3	100	100	98	98
C1_ai	cat3-frame				0.0	0.0				−0.5	100	100	96	96
C1_ai	overall				0.0	0.0	−0.6	−0.5	−0.5	−0.4	100	100	97	97
C2_ai	cat1-A				0.0	0.0	−0.8	−0.8	−0.8		100	100	100	93
C2_ai	cat1-B				0.0	0.0	−0.5	−0.5	−0.5		100	100	99	98
C2_ai	cat3-fused				0.0	0.0	−0.3	−0.5	−0.4	−0.2	100	100	101	97
C2_ai	cat3-frame				0.0	0.0				−0.5	100	100	97	95
C2_ai	overall				0.0	0.0	−0.6	−0.6	−0.6	−0.4	100	100	99	96
CW_ai	cat1-A	100.0	98.1								100	100	94	97
CW_ai	cat1-B	100.0	97.8								100	100	96	97
CW_ai	cat3-fused	100.0	98.0	98.1							100	100	99	89
CW_ai	cat3-frame	100.0		98.0							100	100	104	104
CW_ai	overall	100.0	97.9	98.1							100	100	96	97
CY_ai	cat1-A				0.0	0.0	0.1	0.1	0.1		100	100	100	99
CY_ai	cat1-B				0.0	0.0	−0.3	−0.3	−0.3		100	100	95	97
CY_ai	cat3-fused				0.0	0.0	−1.1	−1.1	−1.1	−0.5	100	100	98	96
CY_ai	cat3-frame				0.0	0.0				−0.5	100	100	96	96
CY_ai	overall				0.0	0.0	−0.2	−0.2	−0.2	−0.5	100	100	97	97

NOTE — Condition CY metrics reported using Hausdorff PSNR.

Table 4 – Performance of proposed coefficient sign coding using RAHT attribute coding compared to v9.0 anchor

		BPP Ratio [%]			BD-Rate [Δ%]						Avg. of ratio maxrssk [%]		Ratio of avg. runtime [%]	
Condition	Class	Geometry	Colour	Refl	D1	D2	Y	Cb	Cr	R	Encoder	Decoder	Encoder	Decoder
C1_ai	cat1-A				0.0	0.0	−0.6	−0.6	−0.6		100	101	97	98
C1_ai	cat1-B				0.0	0.0	−0.5	−0.5	−0.5		100	100	96	97
C1_ai	cat3-fused				0.0	0.0	−0.5	−0.6	−0.6	−0.4	100	100	97	94
C1_ai	cat3-frame				0.0	0.0				−0.5	100	100	97	98
C1_ai	overall				0.0	0.0	−0.5	−0.5	−0.5	−0.5	100	100	97	97
C2_ai	cat1-A				0.0	0.0	−0.9	−1.0	−0.9		100	101	103	97
C2_ai	cat1-B				0.0	0.0	−0.5	−0.4	−0.3		100	100	97	98
C2_ai	cat3-fused				0.0	0.0	−0.5	−0.6	−0.6	−0.3	100	100	95	98
C2_ai	cat3-frame				0.0	0.0				−0.6	100	100	93	93
C2_ai	overall				0.0	0.0	−0.7	−0.7	−0.6	−0.5	100	100	98	97

References

[1] 3DG, "Common Test Conditions for PCC," ISO/IEC JTC1/SC29/WG11, 129th meeting, Brussels, Tech. Rep. w19084, Jan. 2020.

[2] ——, “G-PCC performance evaluation and anchor results,” ISO/IEC JTC1/SC29/WG11, 129th meeting, Brussels, Tech. Rep. w19086, Jan. 2020.