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/m52515
January 2020, Brussels, Belgium

Source:	Apple Inc.	
Status:	Input document	
Title:	CE13.22 report on predictive geometry coding	
Author(s):	David Flynn	*davidflynn@apple.com*
	Khaled Mammou	*kmammou@apple.com*

Abstract

This document reports on the compression performance of the proposed predictive geometry coder as part of core experiment 13.22 [1, N18902].

Introduction

The proposed predictive occupancy coder [2] represents positions as nodes in a tree. The tree is traversed depth-first, with each node having the following properties:

- A prediction mode (eg delta from parent)
- A residual that is combined with the prediction to generate the position of a single point
- A maximum of three child nodes

Four prediction modes are defined, PCM, DPCM, Linear-2 and Linear-3.

Implementation

Two implementations of the encoder (with a common decoder) are available in the `mpeg128/ce13.22/predgeom-high-latency` and `mpeg128/ce13.22/predgeom-low-latency` branches of the CE repository.

The decoder (and bitstream syntax) is common to both implementations. The only difference is the non-normative encoding method.

In order to make the predictor more extensible and the code more readable, in this implementation, the prediction generation is factored out of the coding/mode decision loop.

The encoder implementation is split into two phases for simplicity. The first phase builds the prediction tree according to the non-normative method selected. The second phase (common to both implementations) then encodes the tree while making the final choice of prediction mode.

High-latency encoder

The high-latency encoder uses a k-d tree of predicted point locations generated by previously encoded points. For each point p_i, the nearest prediction $P(p_n)$ is found and the corresponding relationship between p_n and p_i is recorded. Predictions are then generated using p_i and inserted into the k-d tree. In this manner each point may use any of the preceding $p_0 \ldots p_{i-1}$ points as its parent. The tree is not complete until all points have been analysed.

During porting to TMC13v8.0 a number of issues were detected with nanoflann (the k-d tree implementation used by the test model) [3].

Low-latency encoder

The low-latency encoder does not make use of a k-d tree and codes points in timestamp order (when available). The prediction tree is formed by considering the last 128 points (the search range) that have been coded. Meaning that the maximum delay before a node may be encoded is the size of the search window.

Results

To evaluate the performance of the two model encoders, comparisons are made to the TMC13v8 anchors [4] according to the common test conditions [5].

Table 1 shows the performance of the high-latency encoder. Tables 2 and 3 show the performance of the low-latency encoders with respective search ranges of 512 and 128.

The common test conditions attempt to apply a uniform configuration to all content types. However, some features provide no benefit in terms of compression for some types of content – indeed, they may even be a detriment. To provide a fairer comparison of the decoder performance, a modified anchor is produced with the following configuration options set:[1]

- bitwiseOccupancyCoding: 0
- adjacentChildContextualization: 0
- intra_pred_max_node_size_log2: 0

The results of obtained from the high-latency scheme are re-presented in Table 4 using the modified anchor.

Remarks

While we have reported the performance of the system on the full test set, the method is certainly not intended for use with dense point clouds. It does, however, offer gains on sparse map-like data. Gains associated with LiDaR test sequences are diminished compared to those in the original contribution due to new adoptions in TMC13v8 (QtBt in particular).

Table 1 – Performance of the high-latency encoder compared to TMC13v8.0

		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	0.0		103	97	326	70
C1_ai	cat3-fused						0.0	0.0	0.0	0.0	79	90	150	32
C1_ai	cat3-frame									0.0	41	27	252	48
C1_ai	overall						0.0	0.0	0.0	0.0	87	81	286	60
C2_ai	cat1-A				181.9	183.1	0.0	0.0	0.1		99	82	142	85
C2_ai	cat1-B				66.6	66.9					92	65	166	
C2_ai	cat3-fused				45.9	45.6	0.0	−0.0	0.0	−0.0	100	101	144	62
C2_ai	cat3-frame				23.8	23.6				−0.0	48	27	142	50
C2_ai	overall				105.4	106.0	0.0!	0.0!	0.1!	−0.0	90	69	152	
CW_ai	cat1-A	145.6	100.0								108	97	299	75
CW_ai	cat1-B	108.2									79	58	174	9
CW_ai	cat3-fused	90.9	100.0	100.0							79	85	151	31
CW_ai	cat3-frame	99.3		100.0							41	27	260	41
CW_ai	overall	109.8	100.0!	100.0							86	71	224	27
CY_ai	cat1-A						0.0	0.0	0.0		108	98	296	71
CY_ai	cat3-fused						0.0	0.0	0.0	0.0	79	86	155	25
CY_ai	cat3-frame									−0.0	41	27	285	46
CY_ai	overall						0.0	0.0	0.0	−0.0	91	81	276	58

NOTE — Condition CY metrics reported using Hausdorff PSNR.

[1] NB, disabling the bitwise occupancy coder is a little excessive, so compression performance compared to this configuration should be ignored

Table 2 – Performance of the low-latency encoder (search range = 512) compared to TMC13v8.0

		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	0.0		99	97	162	71
C1_ai	cat3-fused						0.0	0.0	0.0	0.0	82	89	79	32
C1_ai	cat3-frame									0.0	33	26	188	52
C1_ai	overall						0.0	0.0	0.0	0.0	83	81	156	62
C2_ai	cat1-A				213.9	215.2	0.1	0.5	0.0		99	82	115	86
C2_ai	cat1-B				79.8	80.2					98	65	107	
C2_ai	cat3-fused				53.6	53.4	−0.0	−0.0	0.0	−0.0	100	100	112	61
C2_ai	cat3-frame				38.8	38.6				0.0	46	27	121	51
C2_ai	overall				125.9	126.6	0.1!	0.4!	0.0!	0.0	92	69	112	
CW_ai	cat1-A	154.6	100.0								100	97	159	77
CW_ai	cat1-B	109.8									68	58	79	10
CW_ai	cat3-fused	95.1	100.0	100.0							81	88	76	29
CW_ai	cat3-frame	103.7		100.0							33	26	181	43
CW_ai	overall	113.0	100.0!	100.0							77	71	115	28
CY_ai	cat1-A						0.0	0.0	0.0		100	98	150	72
CY_ai	cat3-fused						0.0	0.0	0.0	0.0	81	88	79	25
CY_ai	cat3-frame									−0.0	33	26	212	47
CY_ai	overall						0.0	0.0	0.0	−0.0	84	81	152	60

NOTE — Condition CY metrics reported using Hausdorff PSNR.

Table 3 – Performance of the low-latency encoder (search range = 128) compared to TMC13v8.0

		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	0.0		99	97	145	70
C1_ai	cat3-fused						0.0	0.0	0.0	0.0	81	89	74	30
C1_ai	cat3-frame									0.0	33	26	151	50
C1_ai	overall						0.0	0.0	0.0	0.0	83	81	137	60
C2_ai	cat1-A				244.0	245.7	0.0	0.1	0.2		99	82	113	85
C2_ai	cat1-B				93.5	93.8					98	65	101	
C2_ai	cat3-fused				61.0	60.8	0.0	−0.1	0.0	−0.0	100	99	109	62
C2_ai	cat3-frame				43.1	43.0				0.0	46	27	111	51
C2_ai	overall				144.6	145.3	0.0!	0.1!	0.2!	0.0	92	69	107	
CW_ai	cat1-A	163.5	100.0								100	97	140	79
CW_ai	cat1-B	111.4									68	58	72	10
CW_ai	cat3-fused	98.7	100.0	100.0							81	87	76	28
CW_ai	cat3-frame	104.7		100.0							33	26	155	45
CW_ai	overall	115.5	100.0!	100.0							77	71	104	28
CY_ai	cat1-A						0.0	0.0	0.0		100	98	145	71
CY_ai	cat3-fused						0.0	0.0	0.0	0.0	81	88	73	27
CY_ai	cat3-frame									−0.0	33	26	187	45
CY_ai	overall						0.0	0.0	0.0	−0.0	84	81	144	59

NOTE — Condition CY metrics reported using Hausdorff PSNR.

Table 4 – Performance of the high-latency encoder compared to modified 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	−0.0		103	97	389	87
C1_ai	cat3-fused						−0.0	−0.0	−0.0	−0.0	99	103	244	60
C1_ai	cat3-frame									−0.0	41	27	316	65
C1_ai	overall						−0.0	−0.0	−0.0	−0.0	89	83	356	79
C2_ai	cat1-A				133.0	133.9	0.0	0.0	0.1		99	82	138	98
C2_ai	cat1-B				47.4	47.8					94	68	204	
C2_ai	cat3-fused				31.3	31.2	0.0	−0.0	0.0	−0.0	100	101	150	89
C2_ai	cat3-frame				7.3	7.2				−0.0	48	27	158	72
C2_ai	overall				75.2	75.7	0.0!	−0.0!	0.1!	−0.0	91	70	167	
CW_ai	cat1-A	133.0	100.0								108	97	374	91
CW_ai	cat1-B	98.3									107	77	353	23
CW_ai	cat3-fused	88.8	100.0	100.0							99	99	293	56
CW_ai	cat3-frame	97.0		100.0							40	27	384	66
CW_ai	overall	101.6	100.0!	100.0							99	80	362	48
CY_ai	cat1-A						−0.0	−0.0	−0.0		108	98	361	94
CY_ai	cat3-fused						−0.0	−0.0	−0.0	−0.0	99	99	276	52
CY_ai	cat3-frame									−0.0	40	27	367	65
CY_ai	overall						−0.0	−0.0	−0.0	−0.0	93	83	353	82

NOTE — Condition CY metrics reported using Hausdorff PSNR.

References

[1] 3DG, "CE4FE 13.22 Improvements on tree based geometry coding," ISO/IEC JTC1/SC29/WG11, 128th meeting, Geneva, Tech. Rep. w18902, Oct. 2019.

[2] D. Flynn, A. Tourapis, and K. Mammou, "[G-PCC][New proposal] Predictive Geometry Coding," ISO/IEC JTC1/SC29/WG11, 128th meeting, Geneva, Tech. Rep. m51012, Oct. 2019.

[3] D. Flynn and K. Mammou, "G-PCC: Nanoflann software issues," ISO/IEC JTC1/SC29/WG11, 129th meeting, Brussels, Tech. Rep. m52528, Jan. 2020.

[4] 3DG, "G-PCC performance evaluation and anchor results," ISO/IEC JTC1/SC29/WG11, 128th meeting, Geneva, Tech. Rep. w18885, Oct. 2019.

[5] ——, "Common Test Conditions for PCC," ISO/IEC JTC1/SC29/WG11, 128th meeting, Geneva, Tech. Rep. w18883, Oct. 2019.