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/m53391
April 2020, Online

Source: Apple Inc.

Status: Input document

Title: EE13.8 report on low-latency coding for automotive/mapping applications

Author(s): David Flynn *davidflynn@apple.com*
Khaled Mammou *kmammou@apple.com*

Abstract

EE13.8 is tasked with examining the effects of low-latency coding using both the current G-PCC draft geometry coding method and an alternative predictive tree based method. This report analyses the compression performance associated with each method under different latency conditions.

Summary

The predictive geometry codec [1] operates by coding points using a ternary tree. Each tree node represents a point position. Unlike the octree, there is no defined spatial ordering of points in the predictive tree. Compression is achieved by predicting the spatial position of a node based on the positions of parent nodes.

Table 2 shows the summary performance of the scheme when compared to the current v9 common test conditions [2]. Notably the v9 common test conditions include features that assume and exploit known priors in the cat3-frame data that provides a substantial advantage where the assumption holds. By way of summary comparison, Table **??** shows the performance of the same predictive geometry coding compared to the simplest configuration of G-PCC.

Table 1 – Performance of predictive geometry without low-latency slicing compared to v9 CTC

		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
C2_ai	cat3-fused				39.7	39.4					104	78	446	19
C2_ai	cat3-frame				26.5	26.4					43	21	467	11
C2_ai	overall				30.5	30.3					61	38	461	13
CW_ai	cat3-fused	87.4									83	58	319	9
CW_ai	cat3-frame	117.4									40	17	360	15
CW_ai	overall	107.0									53	29	347	13

Table 2 – Performance of predictive geometry without low-latency slicing compared to a simple configuration of TMC13v9

		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
C2_ai	cat3-fused				20.1	20.0					105	92	644	66
C2_ai	cat3-frame				1.5	1.4					112	77	934	44
C2_ai	overall				7.1	6.9					110	82	836	50
CW_ai	cat3-fused	83.2									84	61	955	39
CW_ai	cat3-frame	88.6									36	15	733	35
CW_ai	overall	87.0									51	29	794	36

Slicing and latency

For the purposes of this evaluation, the concept of end-to-end latency is approximated. Given a list of input points, the end-to-end latency is the maximum distance that a point may be displaced in the list.

A slicing method has been added to TMC13 that generates a new slice every n points. This work evaluates three conditions. A single slice per frame, 512 points per slice, and 1024 points per slice.

Within each slice, the encoders (octree, predgeom) are free to reorder the points as they wish. NB, the slicing constraint as implemented here is more strict than necessary for the predictive geometry scheme (however this aspect is not explored further).

The predictive geometry scheme also contains an ability to introduce internal tree boundaries within a slice[1]. This permits the above slicing method to be implemented inside the predictive geometry coder.

Input point ordering

To measure end-to-end system latency, it is necessary to have a defined input point order. While the cat3-frame content is originally acquired by a rotating LiDaR scanner, various processing operations have changed the order of the points in the PLY input.

To restore the input order to something approximating the original acquisition order, a pre-processing stage is added to TMC13 that orders the input point cloud prior to slicing according to the azimuth angle of the points.

While the QNX lidar point clouds are already in an azimuth angle order (the acquisition order), the ford data is not and requires this reordering to be implemented.

Predictive geometry coding

The non-normative tree generation method employed uses a k-d tree to find predicted point locations.

Prior to generating each predictive tree, the input points corresponding to the tree are sorted according to a sorting method[2]. This helps to guide the tree construction process to build a more efficient tree. The sorting methods available are none, morton order, azimuth angle order, and radial distance order.

G-PCC reference configurations

All the results presented are relative to a common reference with slicing disabled. In all cases, attribute coding is disabled.

TMC13v9 Simple (v9smpl)

The reference used for all results is a "simple" configuration of TMC13v9, referenced as "v9smpl" in the results. The purpose of this configuration is to provide a baseline anchor of the minimal G-PCC performance at the lowest complexity operating point. It has the following configuration modifications compared to those used to generate the CTC results [2]:

```
implicitQtBtEnabled: 0
neighbourAvailBoundaryLog2: 0
neighbourContextRestriction: 1
bitwiseOccupancyCoding: 0
adjacentChildContextualization: 0
```

[1] config "predGeomTreePtsMax"
[2] config "predGeomSort"

```
intra_pred_max_node_size_log2: 0
planarEnabled: 0
planarModeIdcmUse: 0
angularEnabled: 0
planarBufferDisabled: 0
```

TMC13v9 LiDaR (v9lidr)

A second configuration that enables features that improve compression in the cat3-frame sequences is referenced as "v9lidr". It has the following modifications compared to the CTC:

```
neighbourAvailBoundaryLog2: 0
neighbourContextRestriction: 1
adjacentChildContextualization: 0
intra_pred_max_node_size_log2: 0
```

The purpose of these modifications is to disable expensive tools that are not expected to contribute significant compression gain, but introduce a significant cost.

TMC13v9 CTC (v9ctc)

A third configuration, referenced as "v9ctc" is identical to the common test conditions configuration in order to provide a comparison to a well known operating point.

Software

This analysis makes use of the following branches of the CE repository:

- `mpeg129/ee13.8/predgeom` — The predictive geometry coder.
- `mpeg129/ee13.8/predgeom-v9ref` — The modified TMC13 anchor.

Both branches include the following:

- Fixes to nanoflann, used by the predictive geometry scheme.
- The n-point slice partitioning method.
- Configuration snippets to configure the slicing modes.
- Pre-sorting of the input point cloud by azimuth angle, including the following fixes:
  - Maintaining point order during duplicate point removal (affects lossy geometry conditions).
  - Using a stable sort criteria to ensure reproducible results.

The predictive geometry coder branch includes the following:

- The previously studied predictive geometry coder[3], with the addition of:
  - a hard bound on the size of any predictive tree (equivalent to the worst case latency due to point reordering), and
  - customisable (non-normative) sorting of points in each predictive tree. The sorting methods are identical to those of the original input contribution, however, the origin used for sorting may differ.
- Duplicate point coding[4]. Including a fix that affected attribute coding (where all duplicate points would have the same attribute value).

The following configuration snippets are provided:

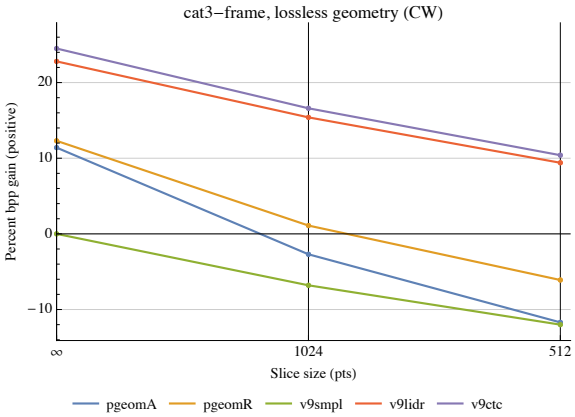


Figure 1 – Compression performance of lossless geometry coding with varying slice sizes of cat3-frame content compared to v9smpl with a single slice per frame

- `cfg-slice=0|512|1024.yaml` partitions the input point cloud into n-point slices in input point order ($n = 0$ disables slicing). NB: the ford data is sorted by azimuth angle to account for the ordering in the ply file (the QNX data is already ordered like this).
- `cfg-predgeom-base.yaml` disables features that are unrelated to the predictive geometry coder.
- `cfg-predgeom=morton|azimuth|radius.yaml` selects the order by which the predictive tree is constructed. The ordering is per predictive tree.
- `cfg-predgeom-tree=512|1024.yaml` selects the maximum number of points per predictive tree (there can be multiple predictive trees per slice).

Analysis

Results are presented to show the performance of both octree coding and predictive tree coding under different slicing conditions.

In the results presented, “pgeomA” corresponds to predictive geometry coding with azimuth angle order, while “pgeomR” corresponds to the radial distance order.

Evaluation of the v9smpl configuration

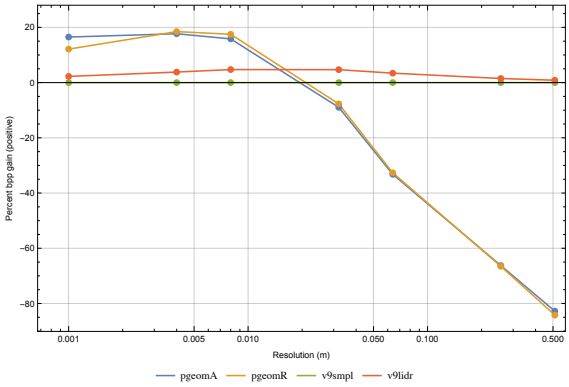
Table 3 describes the performance of the anchor used for all comparisons in this document.

Table 3 – Performance of v9smpl without low-latency slicing compared to v9 CTC

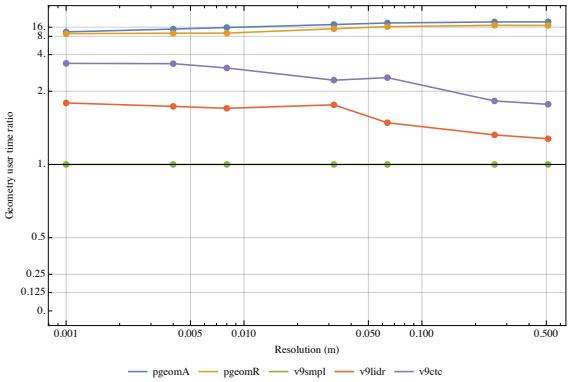
		BPP Ratio [%]			BD-Rate [Δ%]						Avg. of ratio maxrssk [%]		Ratio of avg. runtime [%]	
Condition	Class	Geometry	Colour	Refl	D1	D2	Y	Cb	Cr	R	Encoder	Decoder	Encoder	Decoder
C2_ai	cat3-fused				16.4	16.4					99	86	69	28
C2_ai	cat3-frame				24.7	24.8					38	29	50	25
C2_ai	overall				22.2	22.2					56	46	55	26
CW_ai	cat3-fused	105.1									100	95	33	24
CW_ai	cat3-frame	132.4									111	112	49	44
CW_ai	overall	122.9									108	107	44	36

Effect of slicing on lossless geometry coding

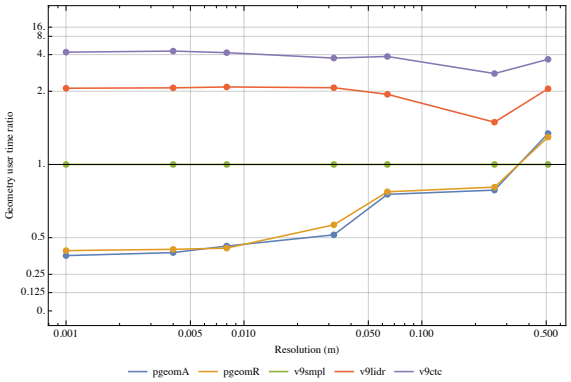
To illustrate the effects of slicing, Figure 1 provides an overview of the lossless geometry coding performance of the various geometry coder configurations when evaluated with different slice sizes. All compression gains are relative to the v9smpl configuration with a single slice per frame.



(a) cat3-fused, single slice/frame, compression gain



(b) cat3-fused, single slice/frame, encode time ratio



(c) cat3-fused, single slice/frame, decode time ratio

Figure 2 – Performance of lossless geometry coding at varying point cloud resolutions compared to v9smpl with a single slice per frame

Effect of resolution on lossless geometry coding

Since all the evaluated configurations employ a lossless geometry codec with quantisation performed as a preprocessing step, the reconstructed point clouds are identical for a given rate point. It is therefore possible to directly compare the compression ratios of individual rate points of the CTC lossy geometry condition.

The performance on cat3-fused content is shown in Figure 2 without any additional slicing beyond the CTC defaults since there is no inherent transmission order of the map content from which to assess a low latency condition.

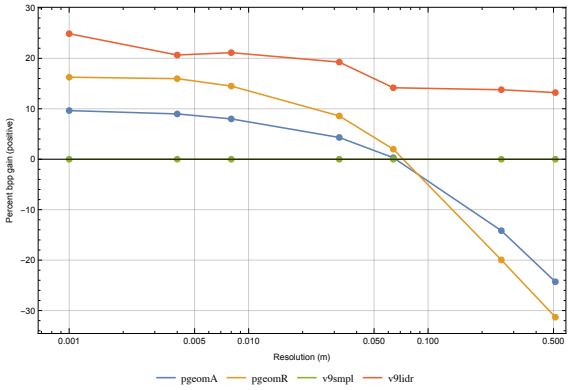
Figures 3 and 4 respectively show the compression and runtime performance of the geometry coders with varying source resolution on cat3-ford and cat3-frame sequences for each latency condition.

References

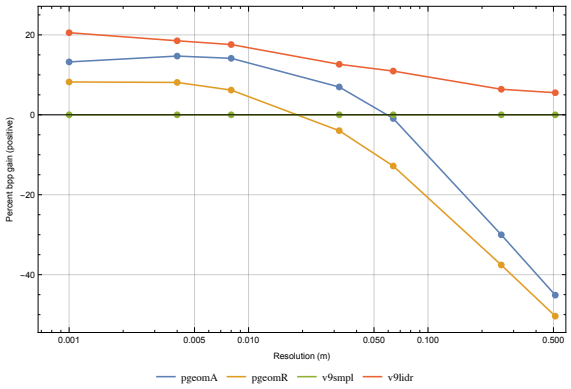
[1] D. Flynn, A. Tourapis, and K. Mammou, “[G-PCC][New proposal] Predictive Geometry Coding,” ISO/IEC JTC1/SC29/WG11, 128th meeting, Geneva, Tech. Rep. m51012, Oct. 2019.

[2] 3DG, “G-PCC performance evaluation and anchor results,” ISO/IEC JTC1/SC29/WG11, 129th meeting, Brussels, Tech. Rep. w19086, Jan. 2020.

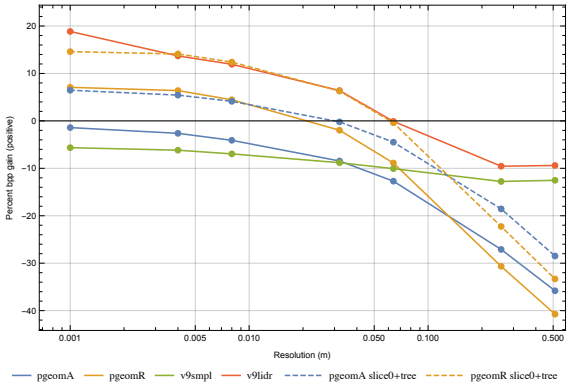
[3] D. Flynn and K. Mammou, “G-PCC CE13.22 report on predictive geometry coding,” ISO/IEC JTC1/SC29/WG11, 129th meeting, Brussels, Tech. Rep. m52515, Jan. 2020.



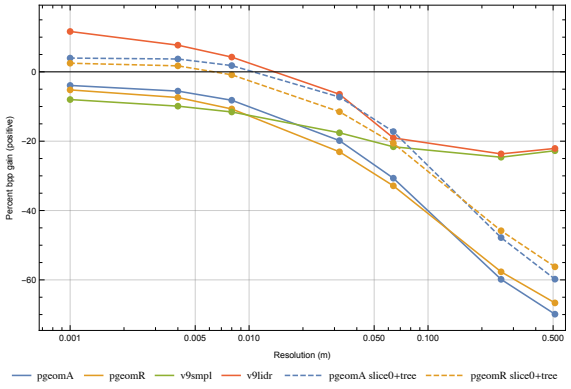
(a) cat3-qnx, single slice/frame, compression gain



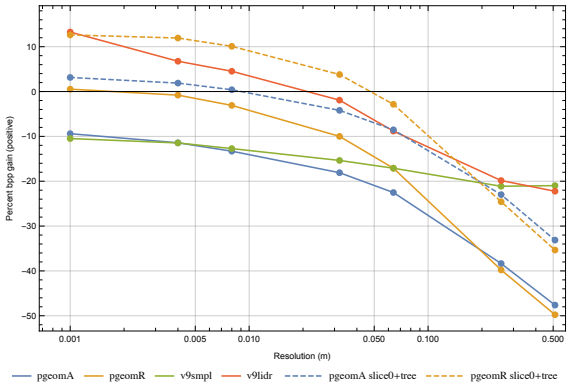
(b) cat3-ford, single slice/frame, compression gain



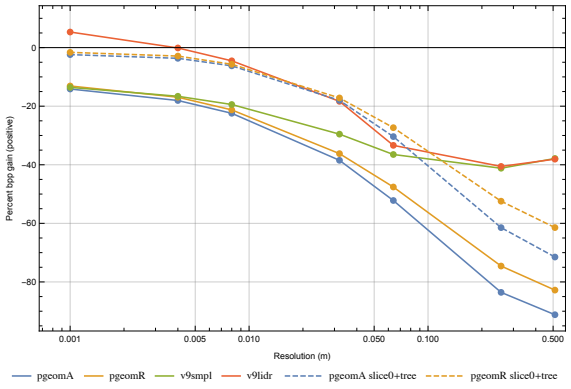
(c) cat3-qnx, 1024 points/slice, compression gain



(d) cat3-ford, 1024 points/slice, compression gain

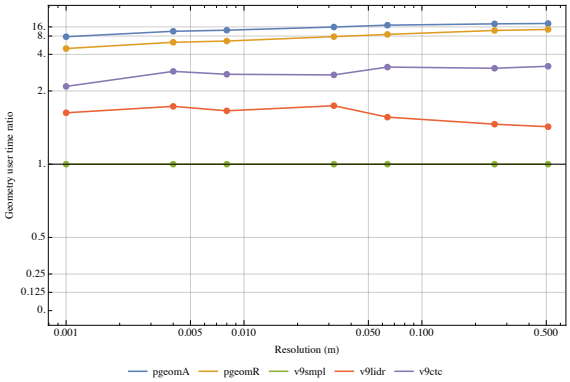


(e) cat3-qnx, 512 points/slice, compression gain

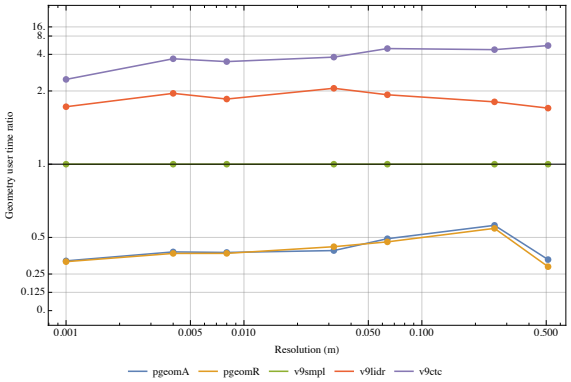


(f) cat3-ford, 512 points/slice, compression gain

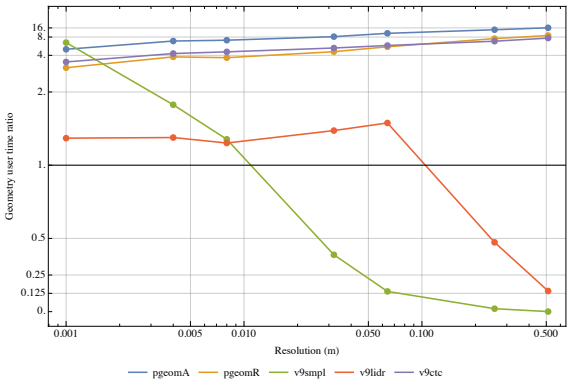
Figure 3 – Performance of lossless geometry coding at varying point cloud resolutions compared to v9smpl with a single slice per frame



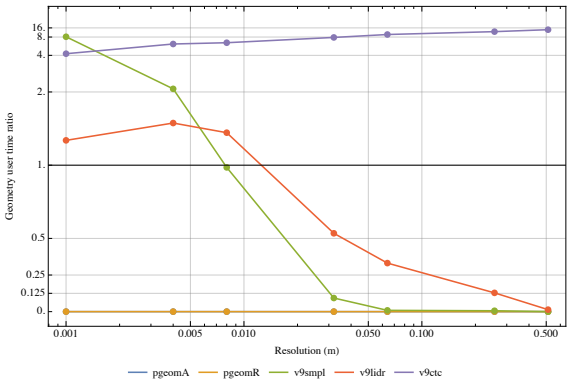
(a) cat3-frame, single slice/frame, encode time ratio



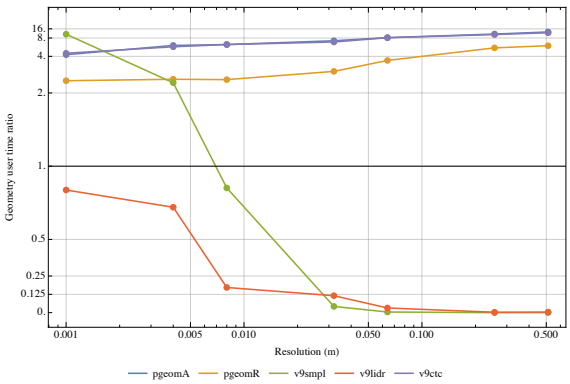
(b) cat3-frame, single slice/frame, decode time ratio



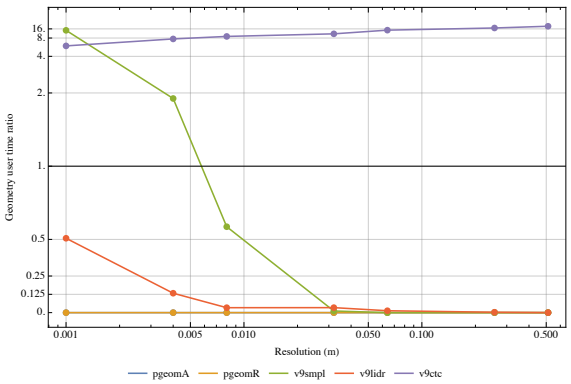
(c) cat3-frame, 1024 points/slice, encode time ratio



(d) cat3-frame, 1024 points/slice, decode time ratio



(e) cat3-frame, 512 points/slice, encode time ratio



(f) cat3-frame, 512 points/slice, decode time ratio

Figure 4 – Performance of lossless geometry coding at varying point cloud resolutions compared to v9smpl with a single slice per frame

[4] ——, “G-PCC: Duplicate point handling in predictive geometry coding,” ISO/IEC JTC1/SC29/WG11, 129th meeting, Brussels, Tech. Rep. m52520, Jan. 2020.