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/m53684
April 2020, Online

Source:	Apple Inc.		
Status:	Input document		
Title:	G-PCC: Review of parallel octree sub-streams		
Author(s):	David Flynn	*davidflynn@apple.com*	
	Khaled Mammou	*kmammou@apple.com*	

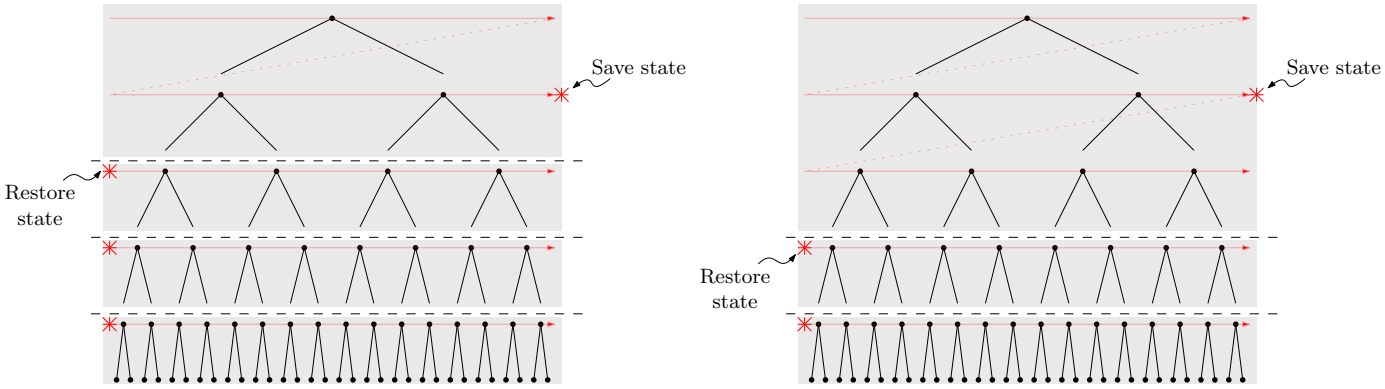
Abstract

At the 129th meeting parallel octree sub-streams were adopted. This contribution reviews their utility and proposes modifications to the high-level syntax.

Introduction

The current draft specification [1] includes a parallelism feature [2] for geometry encoding where in layers of the octree are represented in dependent entropy streams.

At the end of a particular tree depth, the current entropy stream is terminated and context/related state information is saved for use by subsequent dependent streams. All subsequent tree levels are coded as individual dependent streams. This process is illustrated in Figure 1a.



(a) Current (b) Proposed

Figure 1 – Division of octree into multiple entropy streams (grey)

Avoiding confusion between node size and level

The original design was predicated upon using the octree node size to indicate when the first stream terminates. However, other adoptions [3] and other expected adoptions related to qtbt introduce levels where the maximum node dimension is not guaranteed to decrease. In such situations it could be perceived as unclear

as to when the first stream terminates. Furthermore, it is not necessarily intuitive that the node size indicates the level in the octree and how many entropy streams exist.

To resolve this, the draft explicitly indicates the number of parallel streams that exist (minus one, since there must be at least one stream). The determination of when to terminate the first stream is therefore defined based upon the tree level without reference to the node size.

The software currently implements the original method to determine node transitions based upon the maximum node size. We propose to confirm and adopt the method presented in the specification.

Furthermore, the specification is inaccurate since in the determination of GeomEntropyStreamDepth, the value of MaxGeometryOctreeDepth does not take into account any additional octree layers created by qtbt:

```
  MaxGeometryOctreeDepth = gsh_log2_max_nodesize - log2_trisoup_node_size
...
  GeomEntropyStreamDepth = MaxGeometryOctreeDepth - gsh_num_entropy_streams_minusQ - 2
```

The calculation of MaxGeometryOctreeDepth should be updated to be the result of the qtbt derivation process as in the current software.

Removal of a non-parallel stream

As can be seen in Figure 1, the second stream depends upon the entropy state saved at the end of the first stream. It is therefore impossible to generate or decode this stream in parallel with the previous octree level.

This is somewhat acknowledged in the awkward signalling of the number of entropy streams, where the use of two streams is syntactically prohibited.

Here, we propose to simplify the syntax and behaviour by merging the first two streams together. At the end of the designated depth, the entropy state is saved (as per the current design), but the entropy stream termination is deferred until the end of the next octree level (Figure 1b).

```
geometry_data_unit_header() {
  ...
  geom_entropy_streams_minus1 = ue(v)
  if (geom_num_entropy_streams_minus1) {
    geom_entropy_stream_length_bits = u(5)
    /* NB: length of final stream is implicit (no change to v9) */
    for (i = 0; i < geom_num_entropy_streams_minus1; i++)
      geom_entropy_stream_length[i] = u(n)
  }
  ...
```

Utility of signalling offsets

While the entropy state information is known at the start of each dependent stream, it is not possible to entropy decode each stream in parallel since contextualisation is based upon neighbour information present in the previous level.

Unless a parallelism feature is mandated by a profile or level, it is not possible for a decoder design to rely upon its presence. This does not preclude some implementations making opportunistic use of the feature when present. However, the nature of octree node processing makes this non-trivial:

- Due to neighbour dependencies, it is necessary to maintain synchronisation between decoding of a current layer and its parent. Octree nodes are processed in Morton order, however, neighbours are

determined by spatial adjacency. The difference in the two orders results in variable length stalls in processing.

- Octree node processing typically expands the number of tree nodes per level, unless in a special case of sparse (or partially sparse) data. For every node processed, at least one node is produced for processing by the next level. This difference in rate complicates the scheduling of parallel processing of levels in order to make efficient use of available resources.

As such, given the optionality of the feature and the implementation complexity, it is very difficult to see this process as being exploitable by a decoder.

If the feature only makes sense for an encoder, there is no need to permit random access to the streams for a decoder. Instead of signalling the individual stream lengths, we propose to add a between the octree levels that triggers the parallel stream process.

```
geometry_data_unit_data() {
  depthX = depthY = depthZ = 0
  for (depth = 0; depth < MaxGeometryOctreeDepth; depth++) {
    if (!EntropyStreamsEnabled)
      last_octree_level_in_entropy_stream = ae(v)
    ... /* node processing */ ...
  }
}
```

last_octree_level_in_entropy_stream, when present, equal to 1 indicates that the current octree level is the last level of the first geometry entropy stream. When last_octree_level_in_entropy_stream is equal to 1, the parsing state is memorized at the start of the current octree level.

The variable EntropyStreamsEnabled equal to 1 disables the signalling of last_octree_level_in_entropy_-stream and causes each subsequent tree level to start a new dependent entropy stream:

```
if (!EntropyStreamsEnabled)
  EntropyStreamsEnabled = last_octree_level_in_entropy_stream
```

References

[1] 3DG, "Text of ISO/IEC 23090-9 DIS Geometry-based PCC," ISO/IEC JTC1/SC29/WG11, 129th meeting, Brussels, Tech. Rep. w19088, Jan. 2020.

[2] X. Zhang, W. Gao, and S. Liu, "[G-PCC][new proposal] parallel octree coding for point cloud compression," ISO/IEC JTC1/SC29/WG11, 128th meeting, Geneva, Tech. Rep. m50930, Oct. 2019.

[3] J. Taquet and S. Lasserre, "[G-PCC] [CE13.22] Report on harmonization of angular coding mode and implicit QTBT," ISO/IEC JTC1/SC29/WG11, 129th meeting, Brussels, Tech. Rep. m52343, Jan. 2020.