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/m52521
January 2020, Brussels, Belgium

*Source:* Apple Inc.
*Status:* Input document
*Title:* G-PCC: Minor simplifications and fixes to in-tree geometry quantisation

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

## Abstract

In-tree geometry scaling [1, m49232] provides a means to quantise (encoder) and scale (decoder) geometry positions in a non-uniform manner, even while the coding tree is being constructed.

This contribution proposes minor fixes or adjustments to the high-level syntax of geometry quantisation. Furthermore, a solution is suggested to enable compatibility with non-cubic octree nodes (QtBt) and in-tree geometry quantisation.

## Minor fixes to current scheme

### Required clipping in scaling process

Mischievous input notwithstanding, since the specified scaling operation includes rounding, it is possible that a scaled point lies outside the bounds of the node to which it belongs.

For completeness, a michevious input is able to signal a quantised position within a node that is greater than quantising the greatest node position since the entropy coding does not eliminate this possibility.

To resolve both of these cases, it is proposed to add a clipping stage to the decoding process. This was discovered as part of the CE13.29 [2].

```
@@ -908,9 +908,10 @@ invQuantPosition(int qp, Vec3<uint32_t> quantMasks, const Vec3<int32_t
   int shiftBits = (qp - 4) / 6;
   Vec3<int32_t> recon;
   for (int k = 0; k < 3; k++) {
     int posQuant = pos[k] & (quantMasks[k] >>  shiftBits);
     recon[k] = (pos[k] ^ posQuant) << shiftBits;
-    recon[k] |= quantizer.scale(posQuant);
+    recon[k] |= PCCClip(quantizer.scale(posQuant), 0, quantMasks[k]);
   }

   return recon;
```

### geom_base_qp should be geom_base_qp_minus4

A quantisation parameter less than four is not useful in the current scheme. The syntax should be updated to reflect this.

### Ordering of slice header parameters

The scaling parameters in the geometry slice header should come after the geometry box and the maximum node size. These other parameters provide metadata to higher layers, whereas the quantisation parameters do not.

Recommendation: move scaling parameters to end of slice header.

### Remove distinction between in-tree signalling and per-slice signalling

Is geom_octree_qp_offsets_enabled_flag useful?

This flag enables the signalling of per-node offsets. If disabled but scaling is enabled, then the root node is to be scaled with the slice QP. This contrasts with the case where it is enabled, and geom_octree_qp_offsets_-depth=0, which also allows scaling of the root node, albeit with an additional (pointless) offset. The cost of signalling this case with an offset of zero is negligible. From implementation, the effort to avoid this case does not appear to be worthwhile.

Recommendation: remove the offsets_enabled flag (offsets are always enabled when geometry quantisation is enabled).

### Move geom_octree_qp_offsets_enabled_flag from GSH to GPS

Currently, the enable flag for per-node offset signalling is in the slice header. It seems better suited to the geometry parameter set. If an encoder wishes to avoid the use of offsetting in a particular slice, it may always set the offset depth to the maximum.

Recommendation: (if not removed) move enable flag to GPS

### Missing constraints

There are no constraints on qp values or qp offsets. NodeQPs should be constrained such that quantisation will, at most, produce a single point at (0,0,0) relative to the ScalingNode.

### Don't use effective node size to make decisions on enabling features

Geometry quantisation introduces the concept of an effective node size which represents the quantised node size rather than the unquantised node size. The effective node size is used to control the use of intra occupancy prediction (for contextualisation).

```
// generate intra prediction
if (effectiveNodeMaxDimLog2 < gps.intra_pred_max_node_size_log2) {
  predictGeometryOccupancyIntra(
    occupancyAtlas, node0.pos, atlasShift, &occupancyIsPredicted,
    &occupancyPrediction);
}
```

If we consider the case where quantisation step size is two, this has the effect of removing leaf nodes from the quantised subtree. The top of the quantised subtree is unchanged. However, due to the use of the effective node size in the enabling test, intra occupancy prediction is enabled one level earlier.

We suggest reconsidering this and reverting to the previous behaviour where the feature is not conditionally enabled/disabled within a tree level.

## Flexible quantisation offset node size signalling

The current draft signals the geometry tree level at which scaling (inverse quantisation) occurs using the per-slice geom_octree_qp_offset_depth. This value signalled in the slice header and is used to derive GeomScalingDepth, and later ScalingNodeSizeLog2. In many encoder implementations it is preferable to be able to write out the slice header at the start of the geometry slice in order to avoid additional memory copying or other concatenation tricks. Some encoder implementations may wish to decide on-the-fly when (and if) geometry quantisation should be applied.

Instead of signalling geom_octree_qp_offset_depth we propose —

- To indicate at the start of each tree level if node offsets are present in the current level by means of the flag geom_octree_qp_offsets_present_flag.
- After the flag has been set, it is not signalled for any subsequent tree levels.

This has the effect of signalling a unique depth at which offsets are present. The variable GeomScalingDepth is no longer necessary, and ScalingNodeSizeLog2 is derived from the current node size when the flag is asserted. If geom_octree_qp_offsets_present_flag is not enabled, the signalling is used to determine the scaling node size at which whole-slice offsets should be applied.

The slice signalling is modified as follows:

```
 geometry_slice_data( ) {
   depthX = depthY = depthZ = 0;
+  nodeQpOffsetsSignalled = 0
   for( depth = 0; depth < MaxGeometryOctreeDepth; depth++ ) {
+    if( !nodeQpOffsetsSignalled && geom_qp_offsets_enabled_flag ) {
+      node_qp_offsets_present_flag = ae(v)
+      if( node_qp_offsets_present_flag )
+        nodeQpOffsetsSignalled = 1
+    }
     for( nodeIdx = 0 ... ) {
       ...
       geometry_node(...)
     }
   ...
 }
```

**node_qp_offsets_present_flag** equal to 1 indicates that geom_node_qp_offset_eq0_flag is present in each geometry_node of the current tree level. node_qp_offsets_present_flag equal to 0 indicates that geom_-node_qp_offset_eq0_flag is not present in any geometry_node of the current tree level. When not present, geom_node_qp_offset_eq0_flag is inferred to be equal to 0.

```
 geometry_node(...) {
-  if( depth == GeomScalingDepth && geom_octree_qp_offsets_enabled_flag ) {
+  if( node_qp_offsets_present_flag ) {
     geom_node_qp_offset_eq0_flag
     if( !geom_node_qp_offset_eq0_flag) {
       geom_node_qp_offset_sign_flag
       geom_node_qp_offset_abs_minus1
     }
   }
   ...
```

## Interaction with QtBt

Non-cubic octrees (using quad and binary tree partitioning) [3] was adopted at the previous meeting. In the original design, every node at a given depth of the has the same node dimensions and splitting arrangement.

This introduction of in-tree geometry quantisation changes this assumption that all nodes in the same level have the same splitting arrangement. In particular, since tree nodes can now terminate early in a component early, the coded occupancy mask (“occupancySkip”) must be recomputed per node:

The following was added during the software integration of QtBt on top of in-tree quantisation:

```
// todo(??): the following needs to be reviewed, it is added to make
// quantisation work with qtbt.
Vec3<int> actualNodeSizeLog2, actualChildSizeLog2;
for (int k = 0; k < 3; k++) {
  actualNodeSizeLog2[k] = std::max(nodeSizeLog2[k], shiftBits);
  actualChildSizeLog2[k] = std::max(childSizeLog2[k], shiftBits);
}
// todo(??): atlasShift may be wrong too
occupancySkip = nonSplitQtBtAxes(actualNodeSizeLog2, actualChildSizeLog2);
```

It was remarked by the software coordinator that atlasShift may be incorrect. Further study has shown that this is indeed the case.

The occupancy map atlas [4] is used to efficiently determine the occupancy of neighbouring nodes. It is constructed from a run of previously coded nodes spatially relevant to the current node. Of note, it takes each node position and stores the associated 8-bit occupancy information in a 1D array addressed by the Morton code determined from the node position. Since the 8-bit occupancy information represents the bottom three bits of the Morton code, storage efficiency is increased by discarding the bottom three bits of the Morton code.

QtBt causes certain nodes to encode fewer than three bits in their position information and necessarily modifies the address generator to behave as if there were three bits to discard. This is directed by a per-level variable "atlasShift".

However, as stated previously, the use of in-tree quantisation can cause the number of bits encoded per node to vary between nodes in the same tree level.

We propose to address this issue by storing the value of occupancySkip as part of the per-node state information. When the atlas is constructed, the correct value of atlasShift is then used, noting that atlasShift is the complement of occupancySkip.

## References

[1] X. Zhang, W. Gao, S. Yea, and S. Liu, "[G-PCC][New proposal] Signaling delta QPs for adaptive geometry quantization in point cloud coding," ISO/IEC JTC1/SC29/WG11, 127th meeting, Gothenburg, Tech. Rep. m49232, Jul. 2019.

[2] D. Flynn and K. Mammou, "G-PCC CE13.29 report on in-loop geometry quantisation," ISO/IEC JTC1/SC29/WG11, 129th meeting, Brussels, Tech. Rep. m52517, Jan. 2020.

[3] X. Zhang, W. Gao, S. Yea, and S. Liu, "[G-PCC][New proposal] Implicit geometry partition for point cloud coding," ISO/IEC JTC1/SC29/WG11, 127th meeting, Gothenburg, Tech. Rep. m49231, Jul. 2019.

[4] K. Mammou, J. Kim, V. Valentin, F. Robinet, A. Tourapis, and Y. Su, "Look ahead cube for efficient neighbours information retrieval in TMC13," ISO/IEC JTC1/SC29/WG11, 123rd meeting, Ljubljana, Tech. Rep. m43591, Jul. 2018.