**INTERNATIONAL ORGANISATION FOR STANDARDISATION**

**ORGANISATION INTERNATIONALE DE NORMALISATION**

**ISO/IEC JTC 1/SC 29/WG 11**

**CODING OF MOVING PICTURES AND AUDIO**

**ISO/IEC JTC 1/SC 29/WG 11 m** **54713**

**Online – July 2020**

|  |  |
| --- | --- |
| **Source:** | **Sony corporation** |
| **Title:** | **[G-PCC] On Trisoup Coding (Trisoup Node Size per Slice Signaling)** |
| **Authors:** | **Alexandre Zaghetto, Danillo B. Graziosi, Ali Tabatabai,  Ohji Nakagami** |

# Introduction

In G-PCC, trisoup node size is indicated in the Geometry Parameter Set (GPS). Additionaly, in Geometry Header, the trisoup node size parameter defined in GPS is used to indicate the remaining parameters for the trisoup coding, such as sampling value and number of unique segments. If it has a value different than 0, this indicates that trisoup will be used, with the tree level defined in GPS. If one wishes to use slices with trisoup, current notation does not allow for the node size to change on a slice basis. In this contribuiton we propose a high-level syntax modification to allow the control of the node size in trisoup coding on a slice basis.

# Proposal

We propose to send an enable flag GPS and then send the node size value in the GDU header. The high-level syntax modifications for the proposal are presented below. Another possibility is to send a base node size in the GPS, a flag that enables a delta offset, and then send the delta in the GDU header. The high-level syntax modifications for this second apporach are in ANNEX.

#### Geometry parameter set syntax

|  |  |
| --- | --- |
| geometry\_parameter\_set( ) { | **Descriptor** |
| **gps\_geom\_parameter\_set\_id** | ue(v) |
| **gps\_seq\_parameter\_set\_id** | ue(v) |
| **gps\_gsh\_box\_log2\_scale\_present\_flag** | u(1) |
| if( !gps\_gsh\_box\_log2\_scale\_present\_flag) |  |
| **gps\_gs\_box\_log2\_scale** | ue(v) |
| **unique\_geometry\_points\_flag** | u(1) |
| **geometry\_planar\_mode\_flag** | u(1) |
| if( geometry\_planar\_mode\_flag ){ |  |
| **geom\_planar\_mode\_th\_idcm** | ue(v) |
| **geom\_planar\_mode\_th**[ 0 ] | ue(v) |
| **geom\_planar\_mode\_th**[ 1 ] | ue(v) |
| **geom\_planar\_mode\_th**[ 2 ] | ue(v) |
| **geometry\_angular\_mode\_flag** | u(1) |
| } |  |
| if( geometry\_angular\_mode\_flag ){ |  |
| for( k = 0; k < 3; k++ ) |  |
| **geom\_angular\_origin\_xyz**[ k ] | se(v) |
| **number\_lasers\_minus1** | ue(v) |
| **laser\_angle**[ 0 ] | se(v) |
| **laser\_correction**[ 0 ] | ue(v) |
| for( i = 1; i <= number\_lasers\_minus1; i++ ) { |  |
| **laser\_angle\_diff**[ i ] | ue(v) |
| **laser\_correction\_diff**[ i ] | se(v) |
| } |  |
| **planar\_buffer\_disabled\_flag** | u(1) |
| } |  |
| **neighbour\_context\_restriction\_flag** | u(1) |
| **inferred\_direct\_coding\_mode\_enabled\_flag** | u(1) |
| **bitwise\_occupancy\_coding\_flag** | u(1) |
| **adjacent\_child\_contextualization\_enabled\_flag** | u(1) |
| **log2\_neighbour\_avail\_boundary** | ue(v) |
| **log2\_intra\_pred\_max\_node\_size** | ue(v) |
| **trisoup\_enabled\_flag** | u(1) |
| **geom\_scaling\_enabled\_flag** | u(1) |
| if( geom\_scaling\_enabled\_flag ) { |  |
| **geom\_base\_qp** | ue(v) |
| **geom\_direct\_coding\_mode\_qp\_offset** | se(v) |
| } |  |
| **geom\_tree\_coded\_axis\_list\_present\_flag** | u(1) |
| **gps\_extension\_flag** | u(1) |
| if( gps\_extension\_flag ) |  |
| while( more\_data\_in\_byte\_stream( ) ) |  |
| **gps\_extension\_data\_flag** | u(1) |
| byte\_alignment( ) |  |
| } |  |

#### Geometry data unit header syntax

|  |  |
| --- | --- |
| geometry\_data\_unit\_ header( ) { | **Descriptor** |
| **gsh\_geometry\_parameter\_set\_id** | ue(v) |
| **gsh\_tile\_id** | ue(v) |
| **gsh\_slice\_id** | ue(v) |
| **frame\_idx** | u(v) |
| if( gps\_gsh\_box\_log2\_scale\_present\_flag ) |  |
| **gsh\_box\_log2\_scale** | ue(v) |
| for( k = 0; k < 3; k++ ) |  |
| **gsh\_box\_origin\_xyz**[ k ] | ue(v) |
| **geom\_tree\_depth\_minus1** | ue(v) |
| if( geom\_tree\_coded\_axis\_list\_present\_flag ) |  |
| for( lvl = 0; lvl <= geom\_tree\_depth\_minus1; lvl++ ) |  |
| for( k = 0; k < 3; k++ ) |  |
| **geom\_tree\_coded\_axis\_flag**[ lvl ][ k ] | u(1) |
| **gsh\_entropy\_stream\_cnt\_minus1** | ue(v) |
| if( gsh\_entropy\_stream\_cnt\_minus1 ) { |  |
| **gsh\_entropy\_stream\_len\_bits** | u(6) |
| for( i = 0; i < gsh\_entropy\_stream\_cnt\_minus1; i++) |  |
| **gsh\_entropy\_stream\_len**[ i ] | u(v) |
| } |  |
| if( geom\_scaling\_enabled\_flag ) { |  |
| **geom\_slice\_qp\_offset** | se(v) |
| **geom\_octree\_qp\_offsets\_depth** | ue(v) |
| } |  |
| if(trisoup\_enabled\_flag ) { |  |
| **log2\_trisoup\_node\_size** | ue(v) |
| **trisoup\_sampling\_value\_minus1** | ue(v) |
| **num\_unique\_segments\_minus1** | ue(v) |
| } |  |
| byte\_alignment( ) |  |
| } |  |

**trisoup\_enabled\_flag** equal to 1 specifies that geometry bitstream may include trisoup coding syntax. trisoup\_enabled\_flag equal to 0 specifies that geometry bitstream includes only octree coding syntax.

When trisoup\_enabled\_flag is 1, it is a requirement of bitstream conformance that:

- inferred\_direct\_coding\_mode\_enabled\_flag must be equal to 0, and

- unique\_geometry\_points\_flag must be equal to 1.

**log2\_trisoup\_node\_size** specifies the variable TrisoupNodeSize as the size of the triangle nodes as follows.

TrisoupNodeSize = 1 << log2\_trisoup\_node\_size - 1

# Results

To verify the use of different node sizes per slice, we encoded redandblack\_viewdep\_vox12 for r01 to r06. In all cases the original trisoup\_node\_size\_log2 value, as specified in the CTC, was kept constant for slices 1, 2 and 3, and for slice 0 it was set to 1. Figure 1 shows the D1 and D2 Geometry PSNR results.

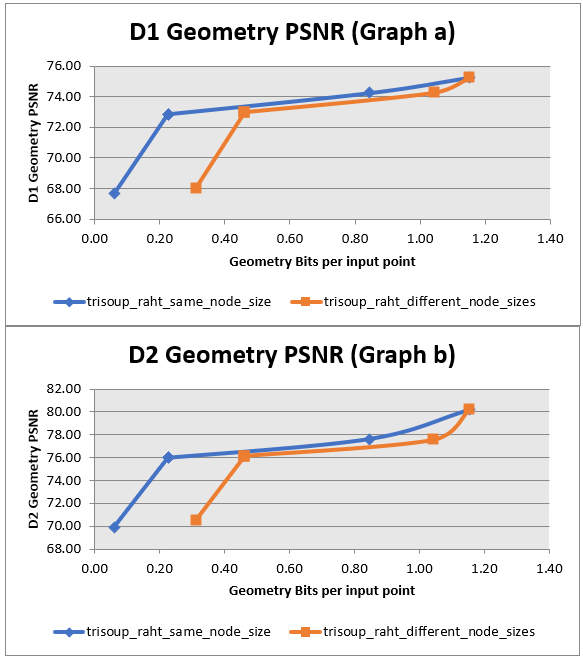


Figure 1. Geometry PSNR plots.

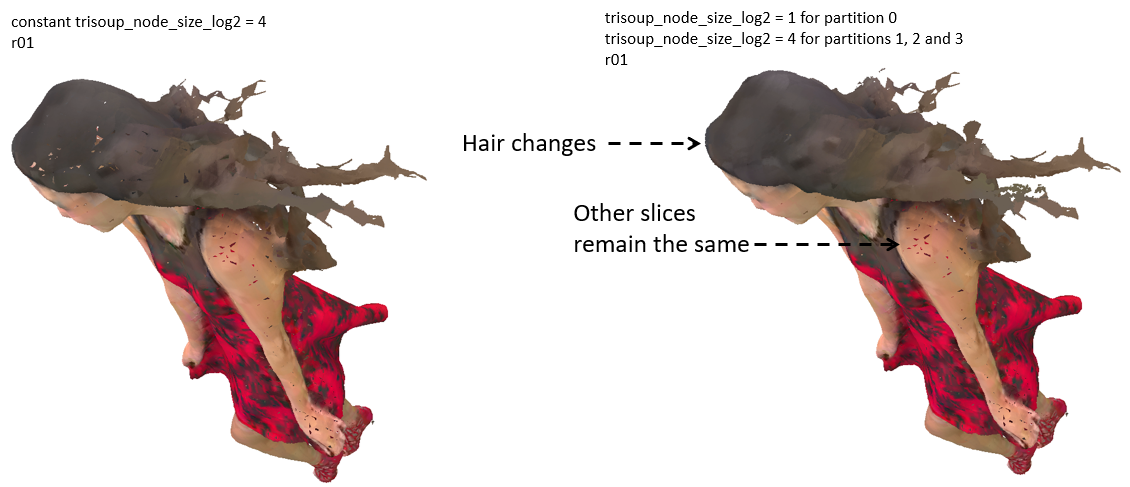
Even though the RD curves show worst performance, by analyzing the subjective quality, we can see that the slice coded with smaller node size has superior quality. This can be efficiently applied in case of ROI encoding. Figure 2 shows an example where redandblack\_viewdep\_vox12 was encoded using trisoup-raht, r04, with (a) trisoup\_node\_size\_log2 = 3 for the whole point cloud; and (b) trisoup\_node\_size\_log2 = 1 for slice 0, and trisoup\_node\_size\_log2 = 3 for the remaining 3 slices. One may notice that slice 0 presents much less artifacts then the following slides.



1. (b)

Figure 2. Example of using different trisoup node sizes for different slices, trisoup-raht, r04.

Figure 3 shows another result for r01 and (a) trisoup\_node\_size\_log2 = 4 for the whole point cloud; and (b) trisoup\_node\_size\_log2 = 1 for slice 0, and trisoup\_node\_size\_log2 = 4 for the remaining 3 slices. One may notice that artifacts in slice 0 are smoothed out from (a) to (b).



1. (b)

Figure 3. Example of using different trisoup node sizes for different slices, trisoup-raht, r01.

# Recommendation

Give the additional flexibility that the proposed high-level syntax modifications introduce into the codec, our recommendation is to introduce the flag in GPS and move the node size to Geometry Data Header.

# ANNEX

Alternative solution that sends a base node size in GPS, a flag that enables a delta offset, and then sends the delta in the GDU header.

#### Geometry parameter set syntax

|  |  |
| --- | --- |
| geometry\_parameter\_set( ) { | **Descriptor** |
| **gps\_geom\_parameter\_set\_id** | ue(v) |
| **gps\_seq\_parameter\_set\_id** | ue(v) |
| **gps\_gsh\_box\_log2\_scale\_present\_flag** | u(1) |
| if( !gps\_gsh\_box\_log2\_scale\_present\_flag) |  |
| **gps\_gs\_box\_log2\_scale** | ue(v) |
| **unique\_geometry\_points\_flag** | u(1) |
| **geometry\_planar\_mode\_flag** | u(1) |
| if( geometry\_planar\_mode\_flag ){ |  |
| **geom\_planar\_mode\_th\_idcm** | ue(v) |
| **geom\_planar\_mode\_th**[ 0 ] | ue(v) |
| **geom\_planar\_mode\_th**[ 1 ] | ue(v) |
| **geom\_planar\_mode\_th**[ 2 ] | ue(v) |
| **geometry\_angular\_mode\_flag** | u(1) |
| } |  |
| if( geometry\_angular\_mode\_flag ){ |  |
| for( k = 0; k < 3; k++ ) |  |
| **geom\_angular\_origin\_xyz**[ k ] | se(v) |
| **number\_lasers\_minus1** | ue(v) |
| **laser\_angle**[ 0 ] | se(v) |
| **laser\_correction**[ 0 ] | ue(v) |
| for( i = 1; i <= number\_lasers\_minus1; i++ ) { |  |
| **laser\_angle\_diff**[ i ] | ue(v) |
| **laser\_correction\_diff**[ i ] | se(v) |
| } |  |
| **planar\_buffer\_disabled\_flag** | u(1) |
| } |  |
| **neighbour\_context\_restriction\_flag** | u(1) |
| **inferred\_direct\_coding\_mode\_enabled\_flag** | u(1) |
| **bitwise\_occupancy\_coding\_flag** | u(1) |
| **adjacent\_child\_contextualization\_enabled\_flag** | u(1) |
| **log2\_neighbour\_avail\_boundary** | ue(v) |
| **log2\_intra\_pred\_max\_node\_size** | ue(v) |
| **log2\_trisoup\_node\_size** | ue(v) |
| **log2\_trisoup\_node\_size\_offset\_present\_flag** | u(1) |
| **geom\_scaling\_enabled\_flag** | u(1) |
| if( geom\_scaling\_enabled\_flag ) { |  |
| **geom\_base\_qp** | ue(v) |
| **geom\_direct\_coding\_mode\_qp\_offset** | se(v) |
| } |  |
| **geom\_tree\_coded\_axis\_list\_present\_flag** | u(1) |
| **gps\_extension\_flag** | u(1) |
| if( gps\_extension\_flag ) |  |
| while( more\_data\_in\_byte\_stream( ) ) |  |
| **gps\_extension\_data\_flag** | u(1) |
| byte\_alignment( ) |  |
| } |  |

#### Geometry data unit header syntax

|  |  |
| --- | --- |
| geometry\_data\_unit\_ header( ) { | **Descriptor** |
| **gsh\_geometry\_parameter\_set\_id** | ue(v) |
| **gsh\_tile\_id** | ue(v) |
| **gsh\_slice\_id** | ue(v) |
| **frame\_idx** | u(v) |
| if( gps\_gsh\_box\_log2\_scale\_present\_flag ) |  |
| **gsh\_box\_log2\_scale** | ue(v) |
| for( k = 0; k < 3; k++ ) |  |
| **gsh\_box\_origin\_xyz**[ k ] | ue(v) |
| **geom\_tree\_depth\_minus1** | ue(v) |
| if( geom\_tree\_coded\_axis\_list\_present\_flag ) |  |
| for( lvl = 0; lvl <= geom\_tree\_depth\_minus1; lvl++ ) |  |
| for( k = 0; k < 3; k++ ) |  |
| **geom\_tree\_coded\_axis\_flag**[ lvl ][ k ] | u(1) |
| **gsh\_entropy\_stream\_cnt\_minus1** | ue(v) |
| if( gsh\_entropy\_stream\_cnt\_minus1 ) { |  |
| **gsh\_entropy\_stream\_len\_bits** | u(6) |
| for( i = 0; i < gsh\_entropy\_stream\_cnt\_minus1; i++) |  |
| **gsh\_entropy\_stream\_len**[ i ] | u(v) |
| } |  |
| if( geom\_scaling\_enabled\_flag ) { |  |
| **geom\_slice\_qp\_offset** | se(v) |
| **geom\_octree\_qp\_offsets\_depth** | ue(v) |
| } |  |
| if(log2\_trisoup\_node\_size ) { |  |
| if(log2\_trisoup\_node\_size\_offset\_present\_flag ) |  |
| **log2\_trisoup\_node\_size\_offset** | se(v) |
| **trisoup\_sampling\_value\_minus1** | ue(v) |
| **num\_unique\_segments\_minus1** | ue(v) |
| } |  |
| byte\_alignment( ) |  |
| } |  |

**log2\_trisoup\_node\_size\_offset\_present\_flag** equal to 1 specifies that trisoup node size offsets indicated by log2\_trisoup\_node\_size\_offset are present in the geometry data unit header.

**log2\_trisoup\_node\_size\_offset\_present\_flag** equal to 0 specifies that no such offsets are present.

**log2\_trisoup\_node\_size\_offset** equal to 1 specifies an offset relative to the log2\_trisoup\_node\_size for use in trisoup coding syntax.

# References

1. “G-PCC Future Enhancements,” ISO/IEC JTC1/SC29 WG11 (MPEG) Document w19328\_d3, Alpbach (Online), CH, January 2020.