**INTERNATIONAL ORGANISATION FOR STANDARDISATION**

**ORGANISATION INTERNATIONALE DE NORMALISATION**

**ISO/IEC JTC1/SC29/WG11**

**CODING OF MOVING PICTURES AND AUDIO**

**ISO/IEC JTC1/SC29/WG11 MPEG2019/m**

**July 2019,** [**Gothenburg**](http://wg11.sc29.org/doc_end_user/current_meeting_intermediate.php?id_meeting=179)**, Sweden**

|  |  |
| --- | --- |
| **Source** | **Peking University Shenzhen Graduate School, Tencent** |
| **Status** | **Input contribution** |
| **Title** | **G-PCC TMC13 CE13.2 report on point cloud tile and slice based coding** |
| **Author** | Yiting Shao([ytshao@pku.edu.cn](mailto:ytshao@pku.edu.cn)), Jiamin Jin, Ge Li, Shan Liu |

# Abstract

This contribution is the report of PCC CE13.2’s development on TMC13v6 [1]. In this CE, we further explore the tile and slice based coding on point cloud.

Upon the current structure on G-PCC, we added a tile partition method with a new parameter *TileSize*, and a new slice partition method based on attribute of each point with two parameters *partitionOctreeDepth, PartitionAttribute*. Additionally, we exchange the order of partition and quantization to ensure the point number of each slice will not be too less in condition C2(lossy geom, lossy attr).

From the aspect of results, we propose to adopt the modified tile and slice partition schemes to the CE13.2 and recommend to add the three new parameters into the CTC.

# Introduction

The goal of this CE is to identify and investigate methods of how to partition point clouds using tile and slice partitioning in TMC13.

A tile is a certain geometry area in the original point cloud, which aims to facilitate spatial random access functionality. And a slice is a set of points that can be decoded independently. Geometry and attribute information of each slice can be encoded/decoded independently. Slice partition in point clouds supports parallelization and other functionalities. Considering the requirement of spatial random access, we would like to get tiles with non-overlapping geometry. So we modified the tile and slice partition order and do the tile partition first. After the tile partition, we get several geometry segmentations, which are only virtual containers without real points, and do slice partition in each tile. Therefore, it still follows the definitions of G-PCC slice and tile.

We have introduced the significance of dynamically adjusting the point number in each slice for different transmission bandwidth in m47227 []. However, at C2(lossy-geom-lossy-attr) condition, though partition can limit the point number of each slice in a required range, quantization will lead to a certain decrease in points, which results in sparseness of some slices and undermined the prediction performance. So we exchange the order of partition and quantization to guarantee the point number of each slice to meet the requirement.

The CE13.2 source code is available on the MPEG Git repository [3]. All experiments are tested with respect to TMC13 v6 under the PCC CTC [2].

# Proposal description

In CE13.2, before geometry and attribute coding on TMC13 v6, a point cloud will be segmented into tiles or slices based on the set of parameters in encode cfg files.

In our contribution, a) the original point cloud would be firstly quantized according to corresponding condition. b) Then the quantized cloud wile be splited into tiles, which are several cube areas with a certain side le*ngth TileSize*. c) After that, the process of slice partition contains two steps. First, do the preliminary slice partition with slice partition schemes in TMC13v6. Second, based on two parameters *MaxPointNum* and *MinPointNum*, do further merging and splitting operations on slices after first step, which try to get slices with suitable number of points. The modification on partition and quantization order could ensure the point number of each slice to meet the requirement.

## Three encode parameters

* *TileSize*: an floating number which defines the side length of a tile.
* *PartitionOctreeDepth*: an integer which defines the depth of octree used by attribute-based partition (default is 1).
* *PartitionAttribute*: Determine which attribute should be used in attribute-base partition

## Quantization modification

Because quantization is moved to the front of partition, there will be a problem that recoloring would not get the original partial point cloud for reference. So we modified the function quantizePositionsUniq() which supports the lossy geometry quantization. The details are showed as follows:

* Letand be the input and the quantized positions, respectively.
* For each point in quantized point cloud, let be the set of points in the original point cloud that share as their quantized position, is the number of elements in , and is one of the elements of Note that is supposed to have one or multiple elements.
* After tile and slice partition, the quantized cloud is splited into several partial cloud .
* Let be the partial cloud in the original point cloud corresponding to . For each point in , find and put them into .
* Use as the reference point cloud to recolor the current slice.

## Tile partition modification

We modified the tile partition as follows:

* Let tileMaps be a map of tile IDs to point indexes that correspond to the tile, initially empty.
* For each point in the reconstructed point cloud after quantization, determine the tile to which it belongs:

1. Let tile\_origin = floor(pos / *TileSize*)
2. Append point index to tileMaps[tile\_origin]

* Then the tile ID is simply the index into the tileMaps
* Do slice partition in each tile.

## Slice partition modification

We have added a new attribute-based partition scheme (kOctreeAttributeOrder: partitionMethod = 4) on the basis of [4]. The details are as follows:

1. Do octree partition at the depth of *PartitionOctreeDepth* to control the loss in geometry prediction. This step will split the input cloud into several large partitions *geoPartitions*.
2. For each partition in *geoPartitions*:

* Get the slice number in current partition:

numSlice = ceil(size() / *MaxPointNum*)

* Sort in the order of a certain attribute value
* Divide into numSlice parts evenly

# Experimental results

The proposed schemes are implemented on TMC13v6. Comparison experiments are tested between the proposed CE13.2 against the TMC13v6 [1] anchor under the updated CTC [2]. The tested conditions are C1 (losslG,lossyA,intra), C2 (lossyG,lossyA,intra), CW(losslG,losslA,intra), CY (losslG,nearllA,intra). The tests were conducted on both Windows and Linux.

There are 6 tests to evaluate the rate-distortion (RD) impacts of the modified tile and slice partition schemes. The details of these tests are the following:

【Tested: octree-predlift + All conditions(C1, C2, CW, CY) + All datasets】

--Test1\_uniform\_tile50000\_maxsize1100000

--Test2\_octreeUniform\_tile50000\_maxsize1100000

【Tested: octree-predlift + All conditions(C1, C2, CW, CY) + Cat1-A, Cat3-fused】

--Test3\_attr\_color\_depth1\_tile50000\_maxsize1100000

【Tested: octree-predlift + All conditions(C1, C2, CW, CY) + Cat3-fused, Cat3-frame】

--Test4\_ attr\_reflectance\_depth1\_tile50000\_maxsize1100000

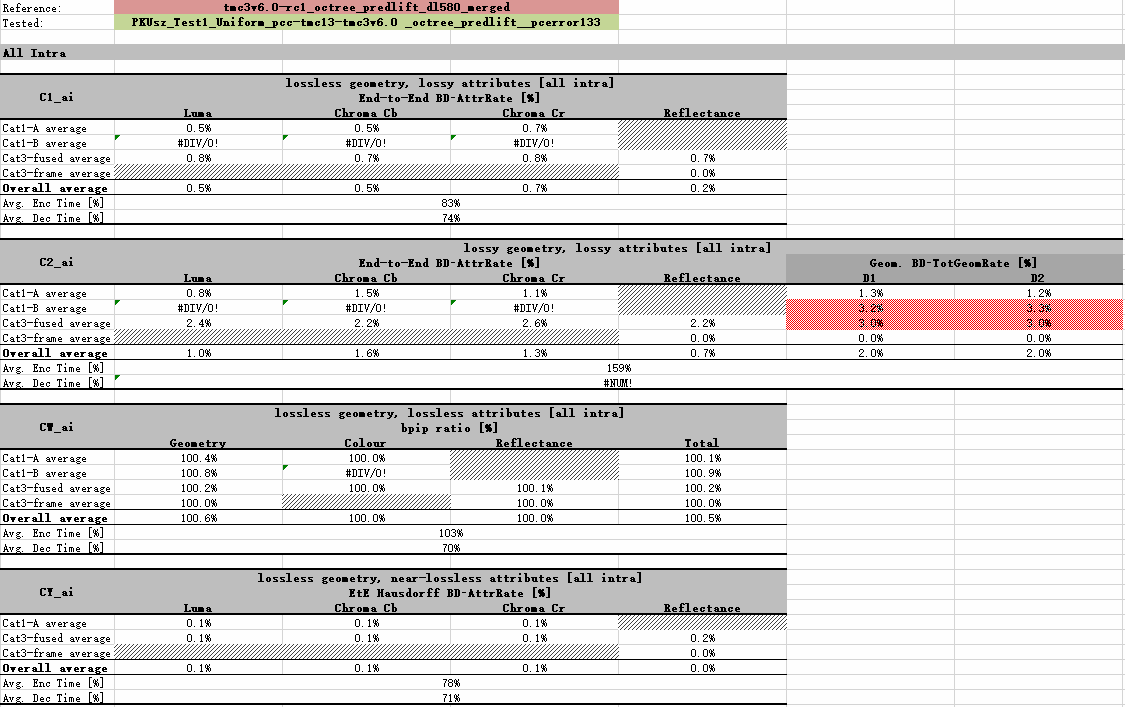
【Contrast Experiments: octree-predlift + All conditions(C1, C2, CW, CY) + All datasets】

--Test5\_uniform\_tile50000\_maxsize1100000\_without\_order\_adjustment

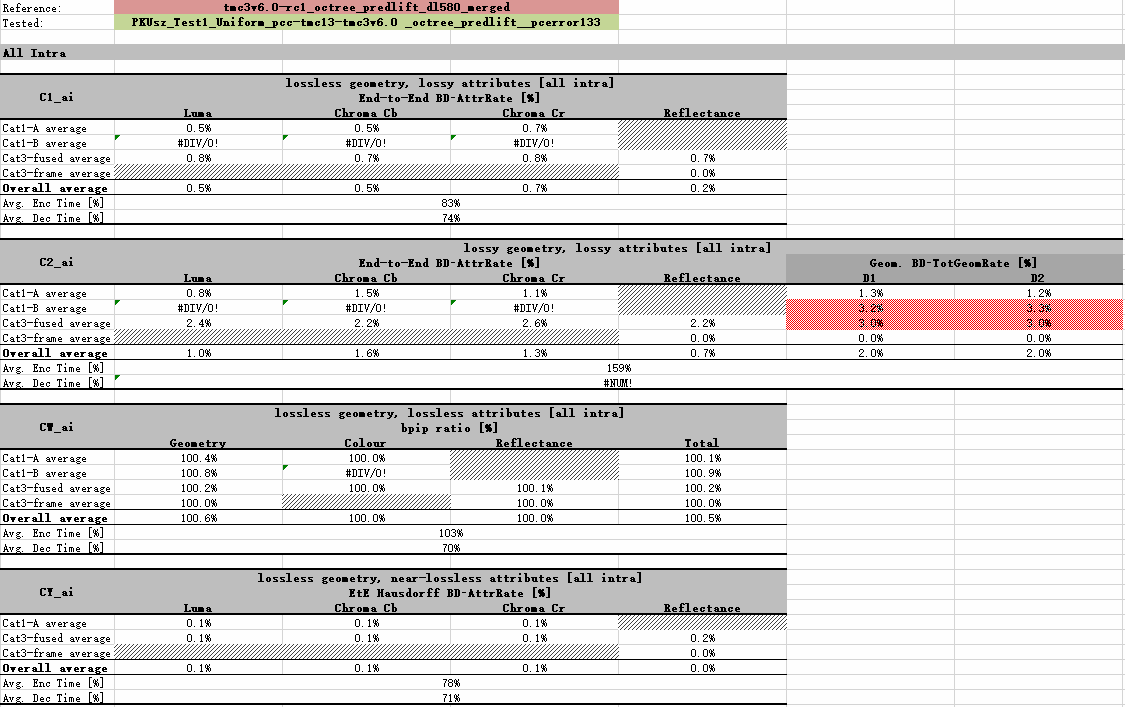
--Test6\_octreeUniform\_tile50000\_maxsize1100000\_without\_order\_adjustment

## RD performances of different slice partition schemes after exchange partition and quantization

### Test1\_uniform\_tile50000\_maxsize1100000

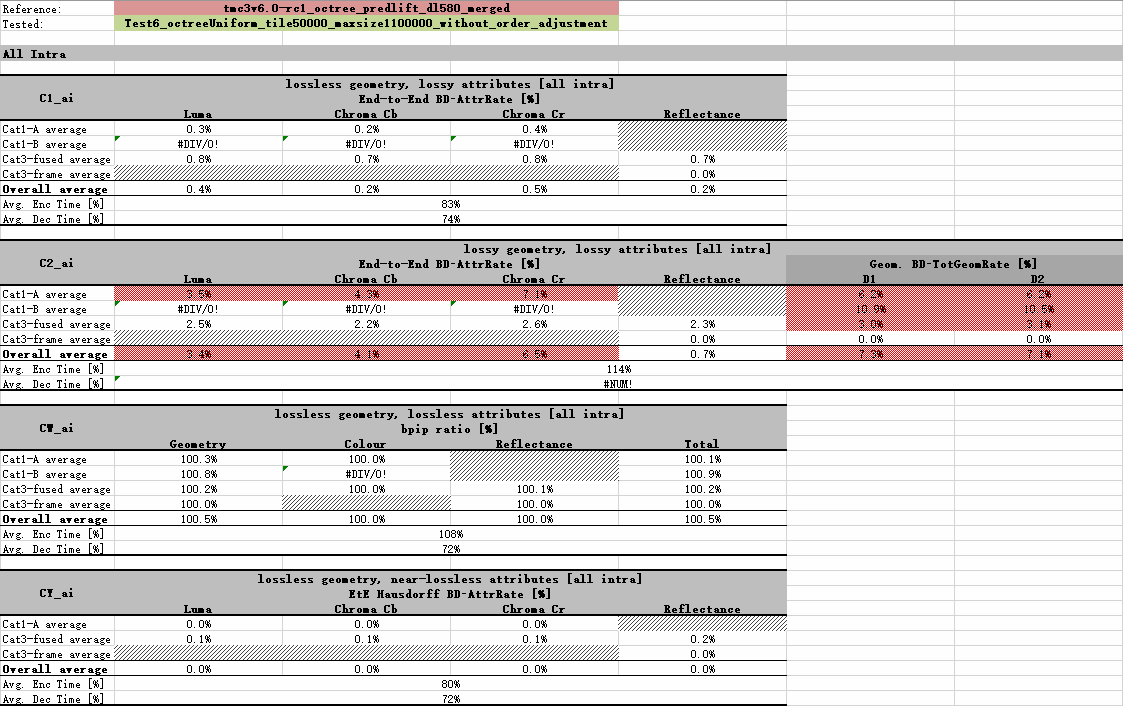


### Test2\_octreeUniform\_tile50000\_maxsize1100000

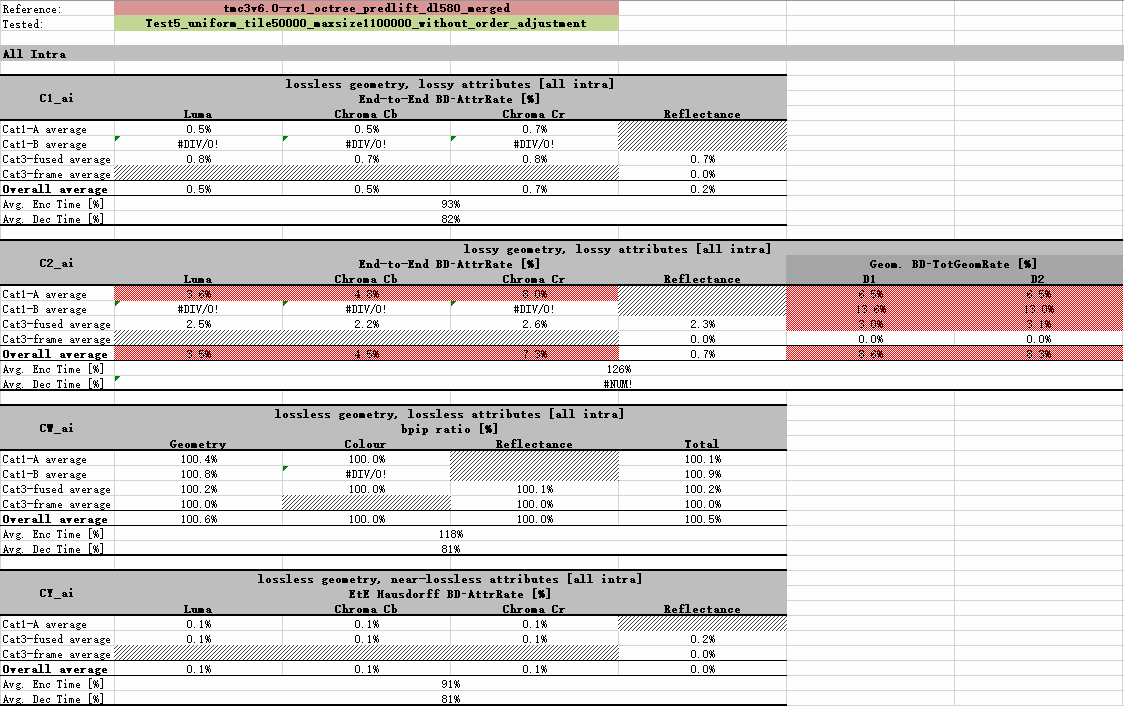


## RD performances of Contrast experiments on the corresponding slice partition scheme without adjust the order of partition and quantization

### Test5\_uniform\_tile50000\_maxsize1100000\_without\_order\_adjustment

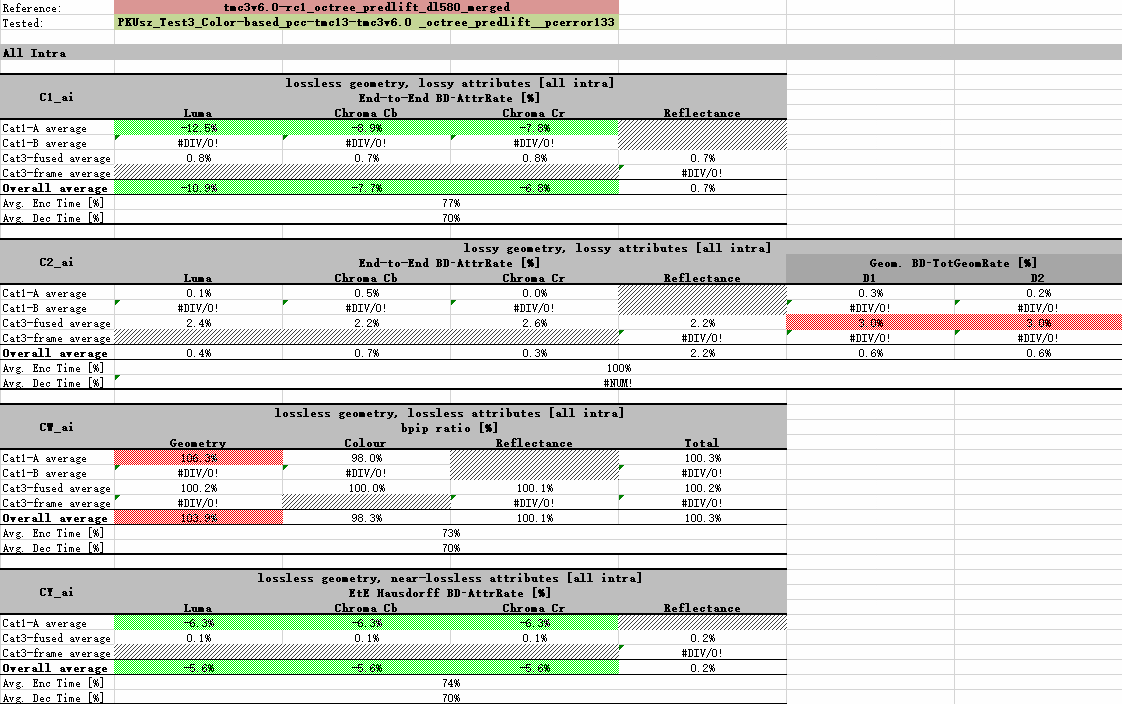


### Test6\_octreeUniform\_tile50000\_maxsize1100000\_without\_order\_adjustment

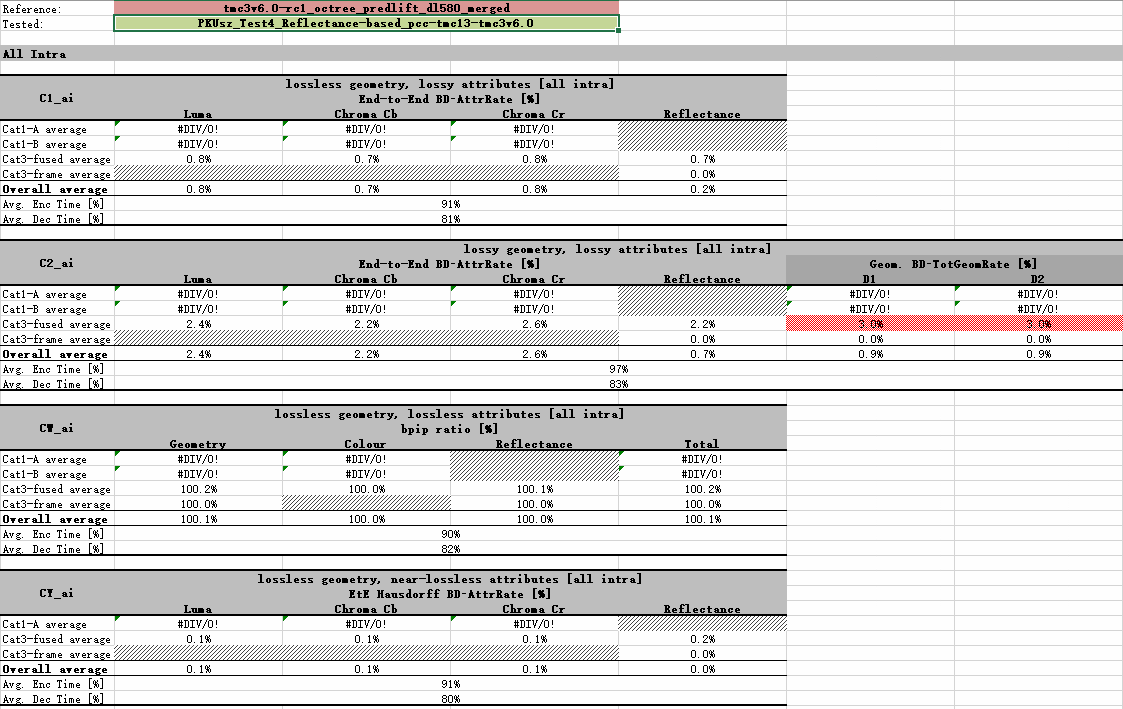


## RD performances of different MaxPointNum and MinPointNum on octree uniform slice partition scheme

### Test3\_attr\_color\_depth1\_tile50000\_maxsize1100000



### Test4\_ attr\_reflectance\_depth1\_tile50000\_maxsize1100000



## Analysis on experimental results

### The impact on RD performance of exchanging the order of partition and quantization

We set test-5 and test-6 as contrast experiments which remain the original order of partitioning before quantization. We can notice that the loss of C2 is much bigger than the other. The main reason is that though partition has limited the point number of each slice in a required range, quantization in lossy geometry condition will lead to a certain decrease in point number, which results in the sparseness of some slices and worsen the prediction performance. For test-1 and test-2, we exchanged the order of partition and quantiation. It is obvious that the RD performance has been significantly improved.

### RD performances of attribute-based slice partition

For test-3, the point cloud is partitioned into slices according to color value. We can see there are remarkable improvement in Luma, Cb and Cr, illustrating its significance for color prediction. However, for test-4, reflectance-based partition, there are no obvious improvement in attribute bitrate. The primary reason may be that the reflectance intensity does not vary acutely between adjacent points, so splitting point clouds according to reflectance cannot bring noteworthy improvement for reflectance prediction. Unfortunately, both color-based and reflectance-based partition have a certain loss in total BD-rate. It is because though the partition methods facilitate the prediction of attribute, it also destroy the geometry construction of original point cloud, which undermined the prediction performance of geometry.

# Conclusion

With the adjustment of partition and quantization, we could guarantee a hard limit on the maximum point count per slice and a soft limit on the minimum point count per slice. Attribute-based slice partition scheme could benefit the attribute prediction and introduce gains on color cases. From the aspect of results, we propose to adopt the modification of quantization and tile partition, also regarding the attribute-based scheme as an optional method.

# References

1. “G-PCC Test Model v6”, ISO/IEC JTC1/SC29/WG11 MPEG N18473, Geneva, March 2019.
2. “Common Test Conditions for PCC”, ISO/IEC JTC1/SC29/WG11 MPEG N18474, Geneva, March 2019.
3. http://mpegx.int-evry.fr/software/MPEG/PCC/CE/mpeg-pcc-tmc13/tree/mpeg126/ce13.2/slice-tile-partition
4. Yiting Shao, Jiamin Jin and Ge Li, “G-PCC TMC13 CE13.2 report on point cloud tile and slice based coding”, ISO/IEC JTC1/SC29/WG11 MPEG N18001, Marrakech, January 2019.