**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,** **m51002**

**Oct 2019, Geneva, Switzerland**

|  |  |
| --- | --- |
| **Source** | **Samsung Electronics** |
| **Status** | **Input contribution** |
| **Title** | **[V-PCC] EE2.6 on mesh coding using V-PCC** |
| **Author** | Esmaeil Faramarzi, Keming Cao, Madhukar Budagavi, Hossein Najaf-Zadeh, and Rajan Joshi |

# Abstract

This contribution is a report of EE2.6 on mesh coding. In the 127th MPEG meeting, Samsung proposed an architectural extension to V-PCC to support mesh coding and presented some preliminary results [1]. This contribution presents some additional results using the previous framework and also presents results for a new framework. The results for lossless configuration are presented and compared with the Google Draco mesh codec [2].

# Proposal

## Dense and sparse meshes

In the 127th MPEG meeting, Samsung proposed an architectural extension to V-PCC to support mesh coding and presented some preliminary results [1]. These results were obtained using a test set consiting of dense meshes created from the CTC point clouds using Poisson Surface Reconstruction and 10-bit voxelization.

In this contribution we present additional results for a second test set. This test set consists of sparse meshes created from the Owlii obj mesh files through downscaling, transforming the texture images to vertex colors, and then voxelizing them to 10 bits.

Both the sets have undergone a cleaning operation to remove (up to 0.25%) non-manifold and degenerative faces resulting from the voxelization procedure.

## Updated results for previously proposed architecture

The method proposed in the 127th MPEG meeting [1] encodes geometry and color using V-PCC which projects points in 3D onto 2D regular patches or RAW patches. The mesh connectivity is encoded using the TFAN mesh coding algorithm from the MPEG SC3DMC software [3]. In the state-of-the-art mesh codecs such as Draco and TFAN, mesh connectivity is encoded first and then geometry and color attributes are encoded using the traversal order of encoded connectivity (e.g. using Parallelogram prediction). However, since in V-PCC, the geometry and color are encoded using 2D video codecs, reordering information needs to be sent to the decoder along with the connectivity information.

We present updated results for the method proposed in the 127th MPEG meeting (referred here as TMC2+TFAN\_F1) for both dense and sparse meshes for lossless coding configuration. Apart from the additional results for sparse meshes, the results for Google Draco are updated. After studying the source code of Draco we noticed that Draco uses floating-point precision when the X,Y,Z geometry components are defined as floating numbers in the header of the mesh Ply files. By changing the precision of geometry from floating-point to integer in the PLY header, as shown in Figure 1, a different compression mode, which is much more efficient, is activated.

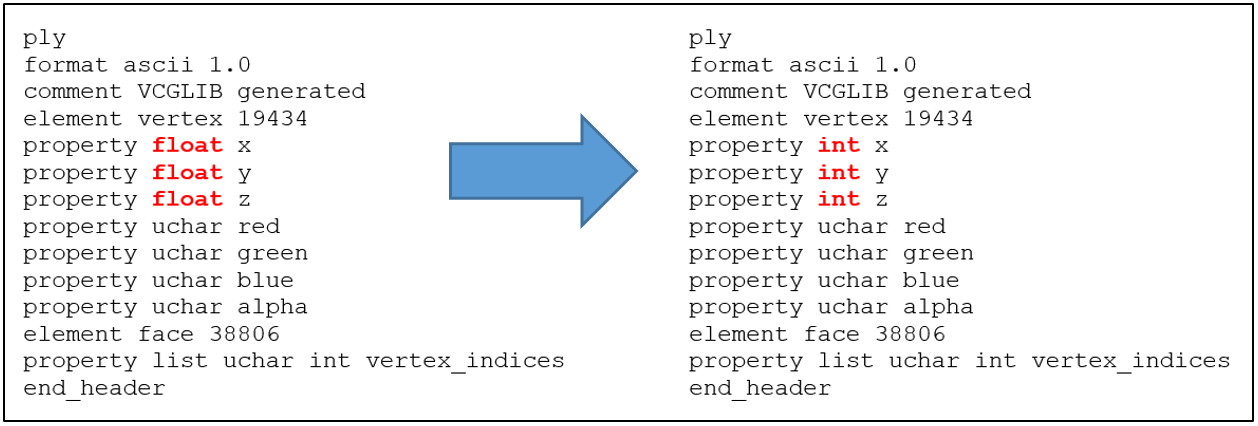


Figure 1: Switching the encoding mode of Draco by changing the precision of geometry from float to int in the mesh PLY header

Another point worth mentioning is that we added print commands in the Draco source code to print out the encoded sizes of individual attributes in the bitstream as Draco only prints out the total encoded size.

It is also our understanding that Draco can't be run in a completely lossless mode. We set the QP and compression level parameters to 0 and 10 (maximum), respectively, to get the highest possible quality.

The lossless coding results for TMC2+TFAN\_F1 and Draco are presented in Tables 1 and 2. Table 3 compares the average performance of TMC2+TFAN\_F1 against Draco (anchor). A negative number (green) in Table 5 indicates better performance (lower bits per point) for TMC2+TFAN\_F1, whereas a positive number (red) indicates better performance for Draco.

It can be seen that for dense meshes, the performance of TMC2+TFAN\_F1 is worse than Draco by an average of 17.29%. However for sparse meshes, TMC2+TFAN\_F1 is worse than Draco by a much bigger margin. To improve the performance of TMC2+TFAN for sparse meshes, we propose a new framework in the next section.

Table 1: Lossless results for TMC2+TFAN\_F1



Table 2: The near-lossless results for Draco



Table 3: Average performance of TMC2+TFAN\_F1 (lossless) versus Draco (anchor, near-lossless)



## Proposed framework for improved performance on sparse meshes

In order to improve the the performance for TMC2+TFAN, we propose a new framework. In this framework, the mesh connectivity is encoded first using TFAN and then geometry and attributes are packed into RAW patches according to the traversal order of TFAN for connectivity. By using this framework, signaling of the reordering information is no longer required. Hence, although the geometry and attribute coding is not as efficient as in the first framework, this is compensated by skipping signaling of the reordering information. Another advantage of the new framework is that it dramatically reduces encoder complexity.

The results for the new method (referred to as TMC2+TFAN\_F2) are presented in Table 4 for lossless configuration. Table 5 compares the average performance of TMC2+TFAN\_F2 vs Draco. It should be noted that when executing our TMC2+TFAN\_F2 code on sparse meshes, the HM decoder (which is 10-bit in TMC2 v5) generates an unmatched md5sum error for the geometry video. Upon investigation, we discovered that only one pixel of the reconstructed geometry frame is different than the input (by one codevalue), whereas the color is completely lossless, as seen in the sample output of the pc\_error metric software for the basketball sequence below. We are trying to track down whether this is a bug in the HM. Due to this one-pixel difference for geometry of sparse meshes, we consider the results for TMC2+TFAN\_F2 to be near-lossless. A negative number (green) in Table 5 indicates better performance (lower bits per point) for TMC2+TFAN\_F2, whereas a positive number (red) indicates better performance for Draco.

3. Final (symmetric).

mseF (p2point): 5.14562e-05

mseF,PSNR (p2point): 107.854

mseF (p2plane): 5.11315e-05

mseF,PSNR (p2plane): 107.882

c[0], F : 0

c[1], F : 0

c[2], F : 0

c[0],PSNRF : inf

c[1],PSNRF : inf

c[2],PSNRF : inf

It can be seen that the results for both dense and sparse meshes show improvement over TMC2+TFAN\_F1. In the case of sparse meshes the improvement is substantial. For both sparse and dense meshes, TMC2+TFAN\_F2 performs slightly worse than Draco (by about 7.5%).

Table 4: Near-lossless results for TMC2+TFAN\_F2



Table 5: Average performance of TMC2+TFAN\_F2 (near-lossless) versus Draco (anchor, near-lossless)



# Conclusion

In this report, we proposed a new framework which eliminated the need for sending reordering information by using RAW patches. In addition to significantly lower encoder complexity, the new framework leads to significantly improved results for sparse meshes. For both sparse and dense meshes the results of the new method are roughly 7.5% worse compared with Draco.

# References

[1] m49588, “[V-PCC] EE2.6 Report on mesh coding,” ISO/IEC JTC1/SC29/WG11, Gothenburg, Sweden, July 2019.

[2] Google Draco mesh codec software, <https://github.com/google/draco>

[3] ISO/IEC JTC 1/SC 29/WG 11, Information technology — Coding of audio-visual objects — Part 16: Animation Framework eXtension (AFX).