INTERNATIONAL ORGANISATION FOR STANDARDISATION
ORGANISATION INTERNATIONALE DE NORMALISATION
ISO/IEC JTC 1/SC 29/WG 7 MPEG CODING FOR 3D GRAPHICS AND HAPTICS

ISO/IEC JTC 1/SC 29/WG 7 m75537
January 2026, Online

Title: [GSC][JEE6.7] On Chroma Subsampling and Upsampling

Authors: Jong-Beom Jeong (KSNU), Jihoon Do, Hahyun Lee, Gun Bang (ETRI)

Abstract

This contribution presents an evaluation of performance variations and experimental results according to different implementation methods for the subsampling and upsampling of spherical harmonics (SH) coefficients converted from RGB to YCbCr.

1 Introduction

As described in [1] at the 152nd MPEG meeting, JEE 6.7 investigates efficient and lightweight video-based solutions for single-frame I-3DGS through the 1F-vid track. Currently, JEE 6.7 operates under three mandates, where Mandate 3 specifically focuses on the investigation of video codec-based technical solutions. This mandate defines exploration across seven sub-topics: quantization, GS-to-2D mapping, 2D map organization, video-codec configurations, SH coefficients, VQ for compression, and the coding of outliers.

Under the video-codec configurations topic, chroma subsampling is addressed, specifically discussing the conversion and subsampling of SH from RGB to YCbCr 4:2:0 format as proposed in [2,3]. Previous works [2,3] proposed a JEE 6.7 Mandate 4 implementation based on the MPEG GSC test Model candidate video (TMCV), confirming that YCbCr conversion and 4:2:0 subsampling of SH yield significant rate-distortion gains. Notably, using the upsampling method specified in subclause B.3.6 of the visual volumetric video coding (V3C, ISO/IEC 23090-5) standard [4] was found to improve visual quality.

This contribution implements and validates the performance variations when SH is converted to YCbCr followed by 4:2:0 subsampling and 4:4:4 upsampling. Versatile video coding (VVC)-based filters were integrated into the TMCV. Furthermore, to enhance local similarity as suggested in [3], tensor preparation was applied to the parallel linear assignment sorting (PLAS) based on the method in [5]. Experimental results show that the proposed method achieves substantial BD-rate gains compared to the conventional simple average subsampling and V3C-based upsampling.

2 Tensor Preparation

TMCV provides options to select specific attributes as criteria prior to the PLAS, with tensor preparation being a key stage. While the original PLAS [6] utilizes position and sh_dc values, the example in [5] converts SH coefficients into RGB for PLAS input. Figure 1 illustrates the source code for tensor preparation in the TMCV: (a) depicts the *integration_152* branch, and (b) depicts the *m73296* branch. The *m73296* branch is shown as it implements the method described in [5].

The SH planes generated using these two methods are shown in Figure 2. The technical differences between these methods were discussed in detail in [3]. Figure 2(b) exhibits higher local similarity than Figure 2(a), resulting in higher coding efficiency in YUV 4:2:0 configurations. Following the analysis in [3], this proposal applies the tensor preparation method from the *m73296* branch.

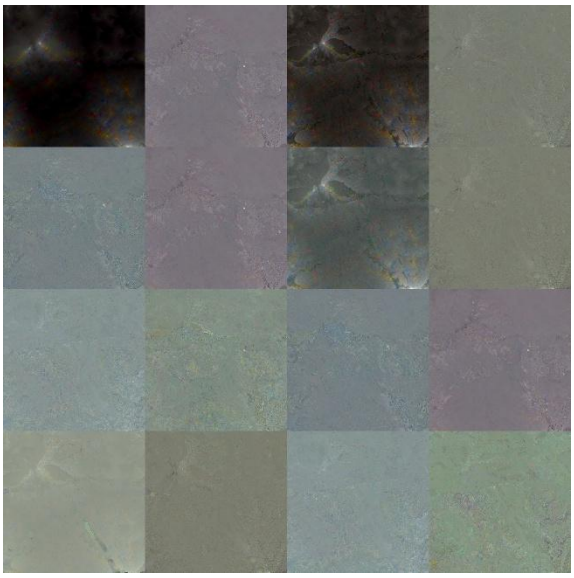
```
tensors = []
for param in param_list:
  if param in ['x', 'y', 'z']:
    values = pointcloud.df[[param]].values
    if trans_position:
      values = log_transform( values )
    norm_values = linear_normalize( values, bitdepth_xyz)
  elif param in ['opacity']:
    values = pointcloud.df[[param]].values
    norm_values = linear_normalize( values, bitdepth_opacity)
  elif param in ['scale_0', 'scale_1', 'scale_2']:
    values = pointcloud.df[[param]].values
    norm_values = linear_normalize( values, bitdepth_scale)
  elif param in ['rot_0', 'rot_1', 'rot_2', 'rot_3']:
    values = pointcloud.df[[param]].values
    norm_values = linear_normalize( values, bitdepth_rotate)
  elif param.startswith('f_dc'):
    dc_vals = pointcloud.df.loc[:, pointcloud.df.columns.str.startswith("f_dc")].values
    norm_values = linear_normalize( dc_vals, bitdepth_dc)
  elif param.startswith('f_rest'):
    values = pointcloud.df[[param]].values
    norm_values = linear_normalize( values, bitdepth_sh)
  else:
    raise ValueError(f"Parameter {param} is not recognized or not handled.")
  tensor_param = torch.from_numpy(norm_values).float().to(device)
  tensors.append(tensor_param)
params_tensor = torch.cat(tensors, dim=1)
return params_tensor
```

(a)

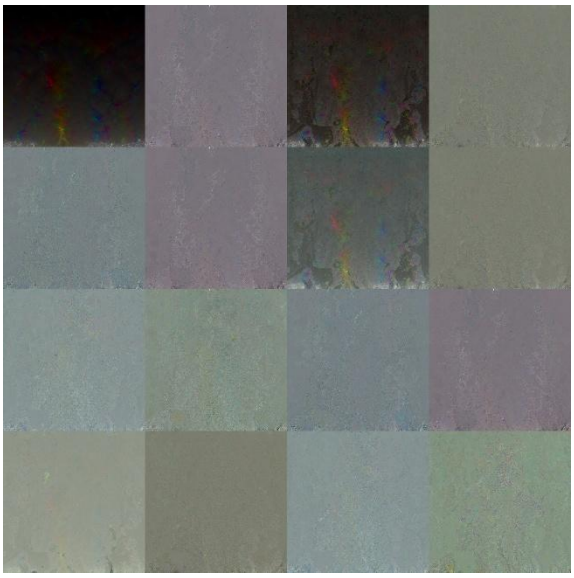
```
tensors = []
COORDS_SCALE = 255
RGB_SCALE = 255
C0 = 0.28209479177387814
df = pointcloud.df
df.sample(frac=1)

coords_xyz = df[["x", "y", "z"]].values
coords_xyz_min = coords_xyz.min()
coords_xyz_range = coords_xyz.max() - coords_xyz_min
coords_xyz_norm = (coords_xyz - coords_xyz_min) / coords_xyz_range
coords_torch = torch.from_numpy(coords_xyz_norm * COORDS_SCALE).float().to(device)

dc_vals = df.loc[:, df.columns.str.startswith("f_dc")].values
rgb = np.clip(dc_vals * C0 + 0.5, 0, 1) * RGB_SCALE
rgb_torch = torch.from_numpy(rgb).float().to(device)

return torch.cat([coords_torch, rgb_torch], dim=1)
```

(b)

Figure 1. TMCV tensor preparation. (a) *integration_152* branch, *m73296* branch



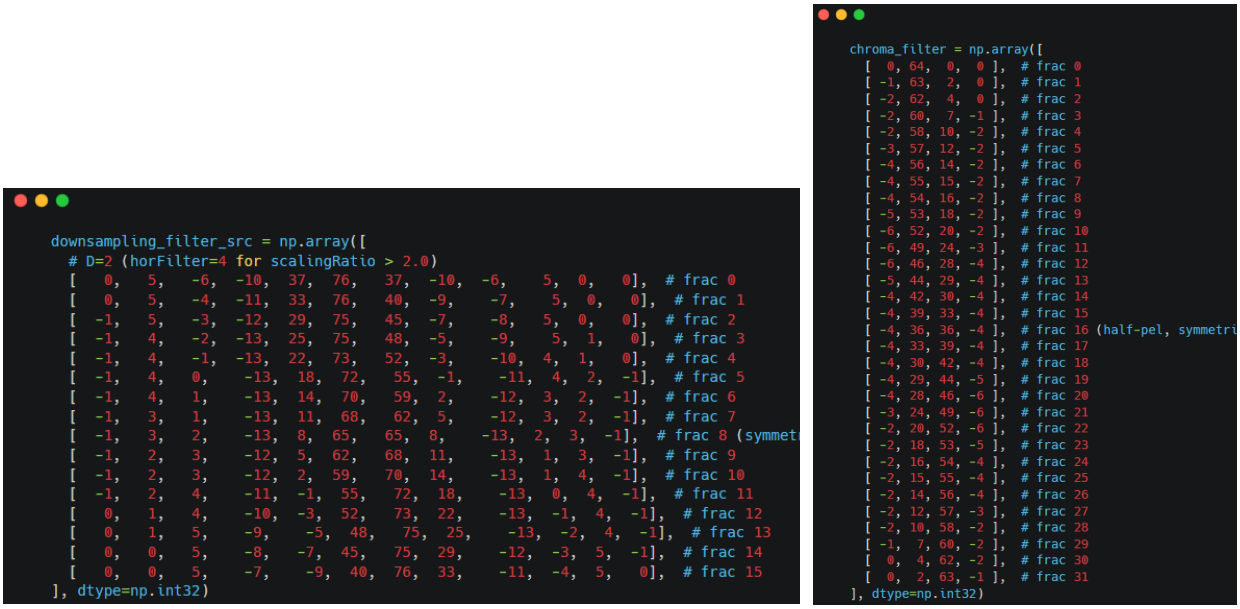
(a)



(b)

Figure 2. SH planes. (a) *integration_152* branch, *m73296* branch

3 Chroma Subsampling and Upsampling



```
downsampling_filter_src = np.array([
  # D=2 (horFilter=4 for scalingRatio > 2.0)
  [   0,   5,   -6,  -10,  37,  76,   37,  -10,  -6,    5,  0,   0],  # frac 0
  [   0,   5,   -4,  -11,  33,  76,   40,  -9,    -7,    5,  0,   0],  # frac 1
  [  -1,   5,   -3,  -12,  29,  75,   45,  -7,    -8,    5,  0,   0],  # frac 2
  [  -1,   4,   -2,  -13,  25,  75,   48,  -5,    -9,    5,  1,   0],  # frac 3
  [  -1,   4,   -1,  -13,  22,  73,   52,  -3,    -10,   4,  1,   0],  # frac 4
  [  -1,   4,   0,    -13,  18,  72,   55,  -1,    -11,   4,  2,  -1],  # frac 5
  [  -1,   4,   1,    -13,  14,  70,   59,  2,     -12,   3,  2,  -1],  # frac 6
  [  -1,   3,   1,    -13,  11,  68,   62,  5,     -12,   3,  2,  -1],  # frac 7
  [  -1,   3,   2,    -13,  8,   65,   65,  8,     -13,   2,  3,  -1],  # frac 8 (symmet
  [  -1,   2,   3,    -12,  5,   62,   68,  11,    -13,   1,  3,  -1],  # frac 9
  [  -1,   2,   3,    -12,  2,   59,   70,  14,    -13,   1,  4,  -1],  # frac 10
  [  -1,   2,   4,    -11,  -1,  55,   72,  18,    -13,   0,  4,  -1],  # frac 11
  [   0,   1,   4,    -10,  -3,  52,   73,  22,    -13,   -1,  4,  -1],  # frac 12
  [   0,   1,   5,    -9,    -5,  48,   75,  25,    -13,   -2,  4,  -1],  # frac 13
  [   0,   0,   5,    -8,   -7,  45,   75,  29,    -12,   -3,  5,  -1],  # frac 14
  [   0,   0,   5,    -7,    -9,  40,   76,  33,    -11,   -4,  5,   0],  # frac 15
], dtype=np.int32)
```

(a)

```
chroma_filter = np.array([
  [  0, 64,  0,  0 ],  # frac 0
  [ -1, 63,  2,  0 ],  # frac 1
  [ -2, 62,  4,  0 ],  # frac 2
  [ -2, 60,  7, -1 ],  # frac 3
  [ -2, 58, 10, -2 ],  # frac 4
  [ -3, 57, 12, -2 ],  # frac 5
  [ -4, 56, 14, -2 ],  # frac 6
  [ -4, 55, 15, -2 ],  # frac 7
  [ -4, 54, 16, -2 ],  # frac 8
  [ -5, 53, 18, -2 ],  # frac 9
  [ -6, 52, 20, -2 ],  # frac 10
  [ -6, 49, 24, -3 ],  # frac 11
  [ -6, 46, 28, -4 ],  # frac 12
  [ -5, 44, 29, -4 ],  # frac 13
  [ -4, 42, 30, -4 ],  # frac 14
  [ -4, 39, 33, -4 ],  # frac 15
  [ -4, 36, 36, -4 ],  # frac 16 (half-pel, symmetri
  [ -4, 33, 39, -4 ],  # frac 17
  [ -4, 30, 42, -4 ],  # frac 18
  [ -4, 29, 44, -5 ],  # frac 19
  [ -4, 28, 46, -6 ],  # frac 20
  [ -3, 24, 49, -6 ],  # frac 21
  [ -2, 20, 52, -6 ],  # frac 22
  [ -2, 18, 53, -5 ],  # frac 23
  [ -2, 16, 54, -4 ],  # frac 24
  [ -2, 15, 55, -4 ],  # frac 25
  [ -2, 14, 56, -4 ],  # frac 26
  [ -2, 12, 57, -3 ],  # frac 27
  [ -2, 10, 58, -2 ],  # frac 28
  [ -1,  7, 60, -2 ],  # frac 29
  [  0,  4, 62, -2 ],  # frac 30
  [  0,  2, 63, -1 ],  # frac 31
], dtype=np.int32)
```

(b)

Figure 3. VTM 2x filter coefficients. (a) subsampling filter, upsampling filter

The chroma subsampling implemented in the TMCV supports two modes: simple drop and average. For upsampling, a Lanczos-2-based filter derived from V3C subclause B.3.6 is utilized. Since this filter operates based on pixels in even rows and columns, it is well-aligned with the simple drop subsampling approach. However, the optimality of these methods for JEE 6.7 requires further investigation. Consequently, this proposal implements the subsampling and upsampling methods from VVC test model (VTM) version 23.13 within the TMCV framework. In VTM, these filters are employed for features such as reference picture resampling (RPR) and are defined in the sampleRateConv() and rescalePicture() functions. For this proposal, the 2x subsampling and upsampling filters were integrated into the TMCV. Figure 3 shows the 2x filters for subsampling and upsampling.

4 Experimental Results and Analysis

4.1 Experimental Conditions

Experiments were conducted on mandatory sequences in accordance with the GSC CTC [7]. The implementation was integrated into the TMCV *integration_152* branch. HEVC test model (HM) version 18.0 served as the video encoder. For rendering and quality assessment, gsTools (*release_mpeg152*) and mpeg-gsc-metrics (*release_2.1*) were utilized [9,10]. Table 1 summarizes the test conditions. The proposed method is designated as Test 2. The difference between Test 1 and Test 3 is the tensor preparation method. All three conditions utilized cfg/hm/ctc/cfg_3_videos.cfg as the base configuration. The first video contains position, and the second video contains opacity, scale, and rotation. The third video contains SH with BT.601 conversion. For all videos, planar packing was applied to enable YUV420 for SH.

Table 2 lists the QP settings employed in these experiments based on [11]. The associated configuration files are attached to this contribution.

Table 1. Test conditions

Name	Tensor Preparation	Subsampling	Upsampling
Test 1	m73296	simple_drop	V3C B.3.6
Test 2	m73296	VTM 12-tap	VTM 4-tap
Test 3	integration_152	simple_drop	V3C B.3.6

Table 2. QP configurations

Rate	QP_1	QP_2
1	-8	0
2	-4	4
3	4	12
4	8	16
5	16	24

4.2 Chroma Subsampling and Upsampling

Table 3 lists the performance results between test 1 and test 2. The anchor employs simple drop and V3C subclause B.3.6-based upsampling. The proposed method applies the VTM-based subsampling and upsampling. The results confirm that the proposed filtering methods yield substantial BD-rate gains without increasing encoding or decoding runtimes.

Table 3. BD-rates, encoding and decoding runtime ratios. Anchor: Test 1, proposed: Test 2. Note that tensor preparation of [3] is applied to both methods.

	RGB-PSNR	YUV-PSNR	YUV-SSIM	Enc. Time (Ratio)	Dec. Time (Ratio)
bartender_ semitracked	-2.9%	-2.8%	-7.9%	100.0%	100.0%
cinema_ semitracked	-5.6%	-5.9%	-7.4%	100.0%	100.0%
breakfast_ semitracked	-3.7%	-3.8%	-4.2%	100.0%	100.0%
breakfast_ untracked	-6.5%	-6.4%	-8.6%	100.0%	100.0%
breakdance_ untracked	-8.6%	-9.0%	-10.6%	100.0%	100.0%
Average	**-5.5%**	**-5.6%**	**-7.8%**	**100.0%**	**100.0%**
manwithfruit_ tracked	-10.8%	-11.1%	-8.5%	100.0%	100.0%

lego_ferrari	-10.0%	-10.8%	-14.7%	100.0%	100.0%
lego_bugatti	-6.5%	-6.8%	-10.1%	100.0%	100.0%
cricket_player	-7.9%	-7.9%	-8.4%	100.0%	100.0%
plant	-4.1%	-4.2%	-6.2%	100.0%	100.0%
solo_tango_ female	-9.3%	-10.4%	-13.0%	100.0%	100.0%
solo_tango_ male	-6.4%	-7.0%	-5.6%	100.0%	100.0%
tango_duo	-11.1%	-11.8%	-14.5%	100.0%	100.0%
tennis_player	-8.9%	-9.6%	-8.6%	100.0%	100.0%
flowerdance	-6.9%	-7.1%	-12.2%	100.0%	100.0%
gymnast	-2.7%	-2.4%	-8.7%	100.0%	100.0%
Average	**-7.7%**	**-8.1%**	**-10.0%**	**100.0%**	**100.0%**

4.3 Tensor Preparation

Table 4 lists the performance results according to the tensor preparation method. While these configurations show further improved BD-rate performance compared to Table 3, an increase in encoding and decoding runtimes was observed. This is attributed to the fact that the YUV components generated by the modified tensor preparation require longer processing times during video coding. The TMCV runtime excluding video codecs showed no significant difference. Future exploration is encouraged to identify methods that enhance BD-rate while minimizing coding complexity.

Table 4. BD-rates, encoding and decoding runtime ratios. Anchor: Test 3, proposed: Test 2. Note that tensor preparation of *integration_152* is applied to the anchor, and [3] is applied to the proposed method.

	RGB-PSNR	YUV-PSNR	YUV-SSIM	Enc. Time (Ratio)	Dec. Time (Ratio)
bartender_ semitracked	-9.4%	-11.3%	-9.7%	357.3%	335.7%
cinema_ semitracked	-10.3%	-10.1%	-11.0%	358.4%	338.4%
breakfast_ semitracked	-4.2%	-4.8%	-4.3%	358.3%	338.6%
breakfast_ untracked	-9.8%	-10.9%	-12.4%	353.7%	331.9%
breakdance_ untracked	2.0%	1.0%	-4.3%	355.7%	337.4%
Average	**-6.4%**	**-7.2%**	**-8.3%**	**356.7%**	**336.4%**
manwithfruit_ tracked	-23.5%	-23.4%	-12.3%	346.0%	401.5%
lego_ferrari	-12.5%	-13.1%	-17.0%	349.4%	346.8%

lego_bugatti	-9.1%	-9.2%	-12.5%	351.0%	349.0%
cricket_player	-11.5%	-11.0%	-7.8%	350.1%	348.1%
plant	-3.7%	-3.6%	-2.4%	349.5%	350.0%
solo_tango_ female	-14.8%	-17.5%	-14.7%	350.5%	348.4%
solo_tango_ male	-8.6%	-9.3%	-2.3%	351.4%	357.5%
tango_duo	-23.2%	-24.6%	-20.4%	348.2%	348.7%
tennis_player	-9.0%	-9.8%	-6.8%	349.7%	351.2%
flowerdance	5.9%	5.4%	-5.9%	351.2%	324.7%
gymnast	19.9%	18.5%	4.4%	352.4%	328.3%
Average	**-8.2%**	**-8.9%**	**-8.9%**	**350.0%**	**350.4%**

4.4 Subjective Evaluations

Figure 4. Enlarged rendered views. First column show source. Second, third, and fourth columns represent rate point 1 (highest), respectively.

Figure 4 shows magnified rendered views for *bartender_semitracked*, *breakfast_semitracked*, and *tango_duo*. The blue and purple distortions prominent in Test 3 are improved in Test 1 and further reduced in Test 2. These results indicate that the improved subsampling and upsampling methods contribute to enhanced subjective quality.

5 Conclusion

This proposal reports performance improvements achieved by improved chroma subsampling and upsampling methods in TMCV. Experimental results confirm that these modifications yield additional BD-rate gains and enhanced subjective quality. Accordingly, it is recommended to crosscheck this in TMCV, integrate into it, and conduct further exploration.

6 References

[1] "JEE 6.7 on coordinating the 1F-vid Track", Standard ISO/IEC JTC1/SC29 WG4, document n0758, October 2025.

[2] J. -B. Jeong, J. Do, H. Lee, G. Bang, "[GSC][JEE6.7] Report of JEE 6.7 Task 3 TT_m73296", Standard ISO/IEC JTC1/SC29 WG4, document m74387, October 2025.

[3] J. -B. Jeong, J. Do, H. Lee, G. Bang, "[GSC][JEE6.7] Report of JEE 6.7 Task 4 Chroma Subsampling", Standard ISO/IEC JTC1/SC29 WG4, document m74388, October 2025.

[4] "Text of ISO/IEC DIS 23090-5 V3C 4th edition", Standard ISO/IEC JTC1/SC29 WG7, document n01154, April 2025.

[5] https://github.com/fraunhoferhhi/PLAS/blob/main/examples/sort_3d_gaussians.py

[6] W. Morgenstern, F. Barthel, A. Hilsmann, P. Eisert, "Compact 3D Scene Representation via Self-Organizing Gaussian Grids", ECCV 2024, pp. 18-34, 2024.

[7] "Common test conditions for Gaussian splat coding", Standard ISO/IEC JTC1/SC29 WG4, document n0751, October 2025.

[8] B. Kroon, P. R. Alface, J. Jung, G. Sandri, "[GSC][JEE 6.8] Proposed CTC changes", Standard ISO/IEC JTC1/SC29 WG7, document m74993, January 2026.

[9] https://git.mpeg.expert/MPEG/Explorations/GSC/gsc-software/mpeg-gsc-tools/-/tree/main/gsTools?ref_type=heads

[10] https://git.mpeg.expert/MPEG/Explorations/GSC/gsc-software/mpeg-gsc-metrics

[11] J. Ricard, G. Teniou, S. Wenger, "[GSC][JEE6.7] TMCV performances and common test conditions", Standard ISO/IEC JTC1/SC29 WG4, document m74517, October 2025.