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 m52532
Brussels, BE – January 2020

Source:	**3DG**
Title:	**[V-PCC][MPEG-I Visual] Sony's Response to CE1.0 on Metadata**
Authors:	**Danillo Graziosi, Alexandre Zaghetto, Ali Tabatabai**

Abstract

This document provides the description of Sony's response to CE1.0 on Metadata. One of the objective's of this Core Experiment is to evaluate the alignment of the MIV activity with the V-PCC specification [1]. In the previous meeting [2], we have demonstrated that MIV content can be encoded using the V-PCC software with some small adaptations in the V-PCC specification. Here we explain the effort to adapt the V-PCC reference software [3] to encode MIV content, and show that, with some minor additions/modifications, the latest version of the V-PCC specification can be used for MIV as well.

1 MIV and V-PCC alignment: patch as a camera-projection.

MIV and V-PCC deal with different representations of a 3D content, one is concerned with the compression of multiple views with associate depth maps, while the other is focused on point clouds. However, both methods utilize segmented projection patches and atlas building to create 2D content from 3D information, and subsequently encode the 2D content with video encoders. Moreover, in a typical V-PCC or MIV bitstream, one would usually find video streams for geometry and texture information, along with metadata to reconstruct the 3D information from the 2D information.

In V-PCC, patches are created by segmenting the point cloud into regions, and orthogonally projecting the points of a particular segmented region to one of the sides of the axis-aligned bounding box. This operation can be interpreted as if an orthographic camera positioned in the origin of the 3D space is capturing part of the point cloud, as shown in Figure 1. Notice that the camea frustum for an orthographic camera is also an axis-aligned bounding box. One can interpret the bounding box of the patch as a volumetric slice of the orthographic camera frustum. The offset (U1,V1,D1) and size (sizeU,sizeV,sizeD) of the bounding box of the patch related to the camera frustum is used to reduce the range of the 3D information. Those values are transmitted as syntax elements of the V-PCC bitstreams. Notice that it is not required that the bounding box tightly surrounds the patch data, and some distance between the data and the bounding box is allowed. This may be used by the V-PCC encoder to more efficiently encode the metadata.

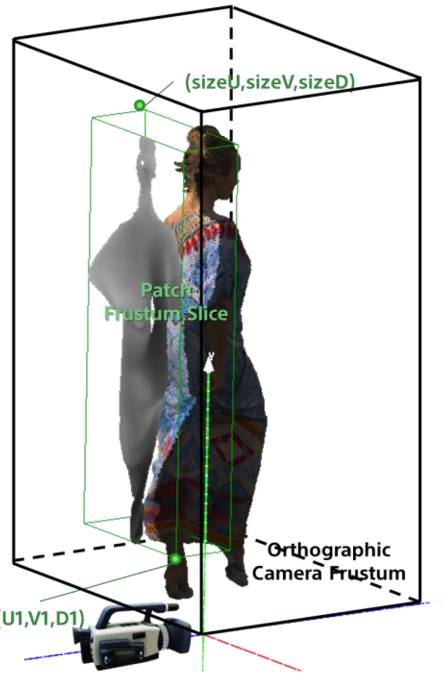

Figure 1: V-PCC orthographic camera model

For the MIV case, the input considered are multiple projected images and their respective depth maps using projective and/or equirectangular camera models. The depth maps can be considered as a 3D information in camera space, because the original 3D values have been transformed by the camera's intrinsic and extrinsic information. If the same camera intrinsic and extrinsic information is used, the depth information can be transformed back to the original 3D information, that is, from camera space to world space. In that case, the information in world space can be interpreted as a point cloud (see Figure 2). Therefore, one could see the MIV content as a point cloud that was already projected to a 2D surface, but using a specific camera model, which is also not restricted to orthographic cameras as with the V-PCC case. In fact, the MIV dataset currently uses perspective and equirectangular cameras.

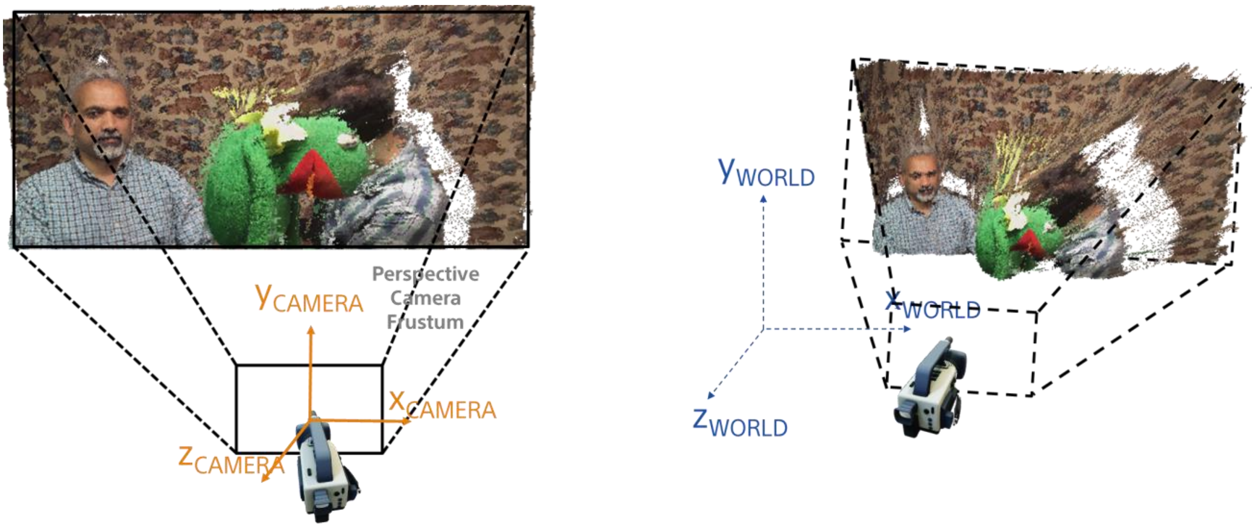

Figure 2: MIV camera model, (left) input in camerea space (right) input in world space

Since V-PCC compresses point clouds as a collection of orthogonal projections of a 3D content, and MIV input can be interpreted as a point cloud that was already projected to a 2D image using either a perspective or equirectangular camera, we believe that the patch concept used in the V-PCC specification can be extended to accommodate both V-PCC and MIV content. The patch can be seen as a volumetric slice of the capturing camera frustum. In the V-PCC case, only orthogonal projections are allowed, as was shown in Figure 1. In the MIV case, only perspective and equirectangular camera models are currently being used. Notice that for MIV, the offset and size of the volumetric patch slice is provided in camera space and at the projection plane, in order to bypass the perspective divide operation necessary to locate the point in 3D space. Since for orthographic cameras there is no perspective divide, one can assume that both MIV and V-PCC are representing the slice offset and size in the projection plane. A visualization of the volumetric slice for perspective cameras can be seen in Figure 3. As an extension to the V-PCC specification, a flexible camera model that includes orthographic, projective and equirectangular models **should** be considered.

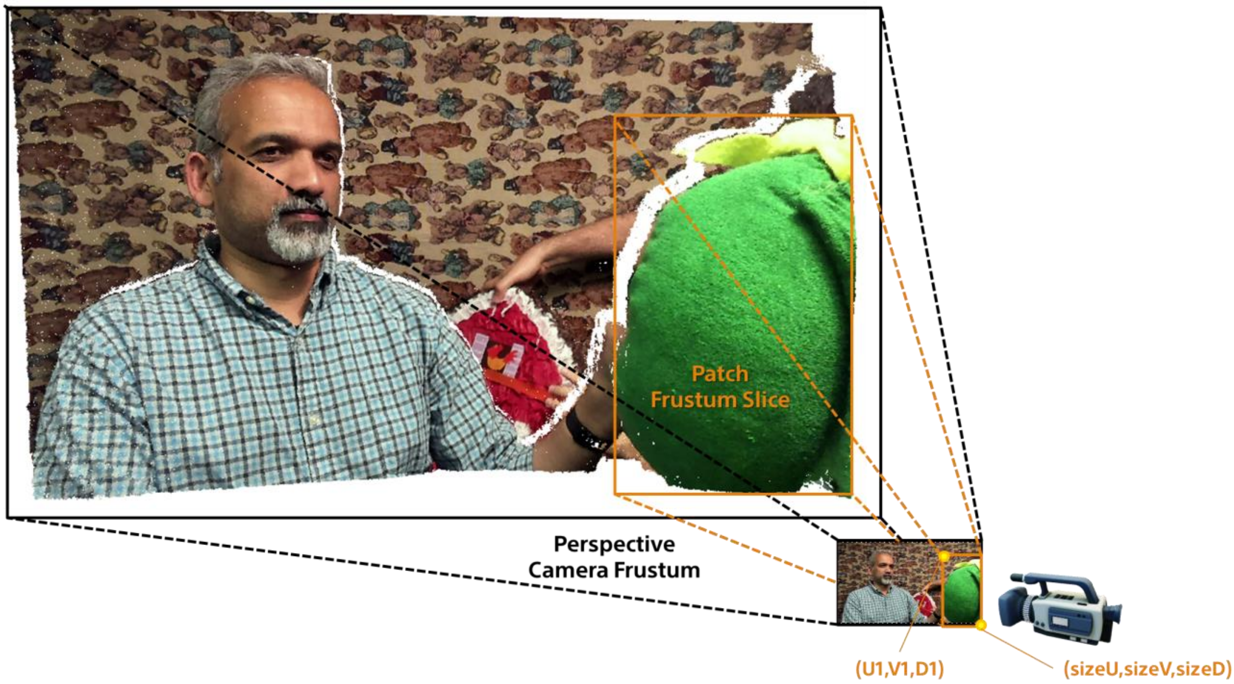

Figure 3: Patch visualization as a frustum slice of perspective camera model.

2 V-PCC specification adaptation for MIV content

In this section we will report the syntax/semantic changes necessary to the V-PCC DIS, to enable the transport of MIV content. Figure 4 shows the syntax elements of a typical MIV bitstream, according to [4].

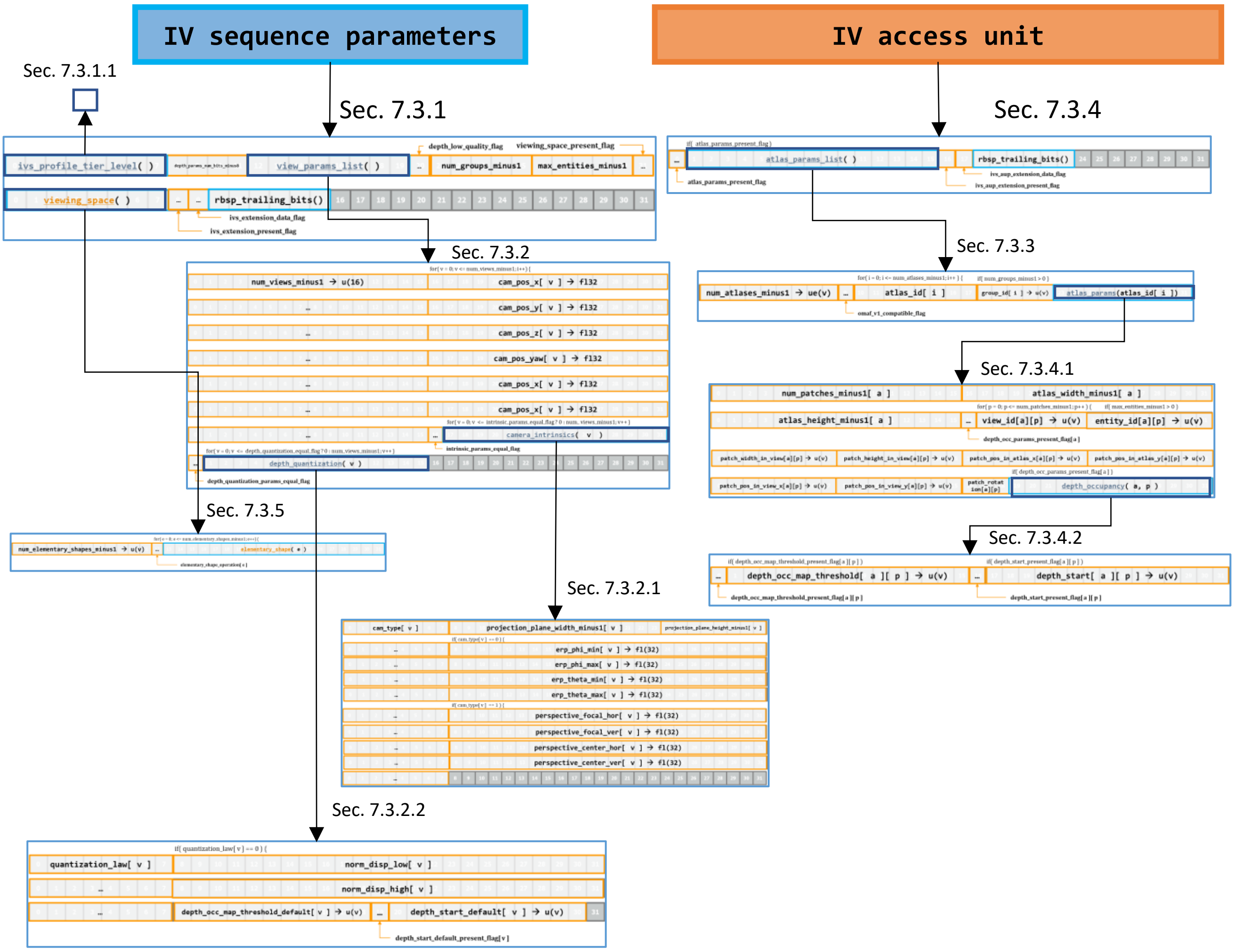

Figure 4: MIV syntax elements

Some of the syntax elements shown above can be directly mapped to existing syntax elements in the V-PCC bitstream, if we consider the patch as a volumetric slice of the "capturing" camera frustum. A table attached to this contribution shows the mapping of the MIV structures to the V-PCC syntax, as implemented in the integration software [7]. A new extension was created to carry some of the MIV syntax elements from the sequence parameter semantics.

2.1 VPS extension for MIV

In the current V-PCC integration software, MIV sequence parameters are being sent using the vps extension mechanism. We propose to extend the VPS in the following manner (new syntax elements in green):

vpcc_parameter_set( ) {	**Descriptor**
profile_tier_level()	

vps_vpcc_parameter_set_id	u(4)
vps_atlas_count_minus1	u(6)
for(j = 0; j < vps_atlas_count_minus1 + 1; j++) {	
vps_frame_width[j]	u(16)
vps_frame_height[j]	u(16)
vps_map_count_minus1[j]	u(4)
if(vps_map_count_minus1[j] > 0)	
vps_multiple_map_streams_present_flag[j]	u(1)
vps_map_absolute_coding_enabled_flag[j][0] = 1	
for(i = 1; i <= vps_map_count_minus1[j]; i++) {	
if(vps_multiple_map_streams_present_flag[j])	
vps_map_absolute_coding_enabled_flag[j][i]	u(1)
else	
vps_map_absolute_coding_enabled_flag[j][i] = 1	
if(vps_map_absolute_coding_enabled_flag[j][i] = = 0) {	
if(i > 0)	
vps_map_predictor_index_diff[j][i]	ue(v)
else	
vps_map_predictor_index_diff[j][i] = 0	
}	
}	
vps_raw_patch_enabled_flag[j]	u(1)
if(vps_raw_patch_enabled_flag[j])	
vps_raw_separate_video_present_flag[j]	u(1)
occupancy_information(j)	
geometry_information(j)	
attribute_information(j)	
}	
vps_miv_sps_present_flag	u(1)
if(vps_miv_sps_present_flag) {	
miv_sps_length_minus1	ue(v)
miv_sequence_parameter_set(miv_sps_legth_minus1)	u(8)
}	
}	
vps_extension_present_flag	u(1)
if(vps_extension_present_flag) {	
vps_extension_length	ue(v)
for(j = 0; j < vps_extension_length + 1; j++) {	
vps_extension_data_byte	u(8)
}	
}	
byte_alignment( )	

}	

Note: Current software implemented only one single extension, which is the MIV extension. Further extensions can be easily added following the same mechanism.

miv_sequence_parameter_set(miv_sps_legth_minus1) {	**Descriptor**
~~ivs_profile_tier_level( )~~	
depth_params_num_bits_minus8	u(4)
view_params_list( )	
depth_low_quality_flag	u(1)
num_groups_minus1	ue(v)
max_entities_minus1	ue(v)
viewing_space_present_flag	u(1)
if(viewing_space_present_flag)	
viewing_space( )	
~~**ivs_sp_extension_present_flag**~~	~~u(1)~~
~~if(ivs_sp_extension_present_flag) {~~	
~~while(more_data_in_payload( ))~~	
~~**ivs_sp_extension_data_flag**~~	~~u(1)~~
~~rbsp_trailing_bits( ) [Ed. (JB): Is this still needed?]~~	
byte_alignment( )	
while(more_data_in_miv_sps)	
trailing_zero_8bits /* equal to 0x00 */	f(8)
}	

Note: Profile information and extension were removed from this syntax element. Profile and tier information could be aded in the profile and tier syntax element from V-PCC, MIV should verify if the syntax elements in V-PCC are enough, or if more should be added. Extension syntax elements were removed. Current integration software implemented the above syntax, but did not include the elements marked with gray background. This information is not required to decode the patches, and could be either included in the vps extension, or sent through SEI messages.

view_params_list( ) {	**Descriptor**
cam_pos_granularity_x	u(32)
cam_pos_granularity_y	u(32)
cam_pos_granularity_z	u(32)
num_views_minus1	u(16)
for(v = 0; v <= num_views_minus1; i++) {	
cam_pos_x[v]	fl(32)
cam_pos_y[v]	fl(32)
cam_pos_z[v]	fl(32)
cam_yaw[v]	fl(32)
cam_pitch[v]	fl(32)
cam_roll[v]	fl(32)
}	
intrinsic_params_equal_flag	u(1)
for(v = 0; v <= intrinsic_params_equal_flag ? 0 : num_ views_minus1; v++)	
camera_intrinsics(v)	
depth_quantization_params_equal_flag	u(1)
for(v = 0; v <= depth_quantization_equal_flag ? 0 : num_views_minus1; v++)	
depth_quantization(v)	
}	

Note: added granularity to voxelize the points in the reconstruction phase. This can be further discussed, to identify the best method, or even if it is necessary to voxelize the reconstructed 3D information.

camera_intrinsics(v) {	**Descriptor**
cam_type[v]	u(8)
projection_plane_width_minus1[v]	u(16)
projection_plane_height_minus1[v]	u(16)
if(cam_type[v] == 0) {	
erp_phi_min[v]	fl(32)
erp_phi_max[v]	fl(32)
erp_theta_min[v]	fl(32)
erp_theta_max[v]	fl(32)
} else if(cam_type[v] = = 1) {	
perspective_focal_hor[v]	fl(32)
perspective_focal_ver[v]	fl(32)
perspective_center_hor[v]	fl(32)
perspective_center_ver[v]	fl(32)
}	
}	

depth_quantization(v) {	**Descriptor**
quantization_law[v]	u(8)
if(quantization_law[v] == 0) {	
norm_disp_low[v]	fl(32)
norm_disp_high[v]	fl(32)
}	
~~**depth_occ_map_threshold_default**[v]~~	~~u(v)~~
~~**depth_start_default_present_flag**[v]~~	~~u(1)~~
~~if(depth_start_default_present_flag[v])~~	
~~**depth_start_default**[v]~~	~~u(v)~~
}	

Note: The depth occlusion map was removed, since it is not being used in the current implementation, and we just assume that all values are occupied. Occupancy map could be transmitted similar to V-PCC, as a separate video stream, or could be embedded in the geometry like proposed by MIV. However, embedding the occupancy map might affect the current V-PCC syntax, and should be further discussed.

2.2 Patch decoding for flexible camera definition

The atlas parameters in the MIV specification can have most of its syntax elements mapped to the patch syntax elements in V-PCC. We advocate for the use of V-PCC's defintion of patch, since the standard also includes more flexible patch definitions, such as skip and inter predicted patches, which could be useful for MIV as well. Table 1 shows the mapping between MIV and V-PCC syntax elements (also availabel in an excel file attached to this contribution).

Table 1:Mapping of MIV atlas parameter syntax elements to V-PCC syntax elements

MIV SYNTAX ELEMENT	DESCRIPTION	V-PCC CORRESPONDING SYNTAX ELEMENTS	DESCRIPTION
7.4.6.1 **Atlas parameters semantics**			
num_patches_minus1[a]	u(16)	NOT PRESENT	
atlas_width_minus1[a]	u(16)	**vps_frame_width**[j]	u(16)
atlas_height_minus1[a]	u(16)	**vps_frame_height**[j]	u(16)
depth_occ_params_present_flag[a]	u(1)	NOT PRESENT	
view_id[a][p]	u(v)	**pdu_projection_id**[patchIdx]	u(v)
entity_id[a][p]	u(v)	NOT PRESENT	
patch_width_in_view_minus1[a][p]	u(v)	**pdu_2d_delta_size_x**[patchIdx]	se(v)
patch_height_in_view_minus1[a][p]	u(v)	**pdu_2d_delta_size_y**[patchIdx]	se(v)
patch_pos_in_atlas_x[a][p]	u(v)	**pdu_2d_pos_x**[patchIdx]	u(v)
patch_pos_in_atlas_y[a][p]	u(v)	**pdu_2d_pos_y**[patchIdx]	u(v)
patch_pos_in_view_x [a][p]	u(v)	**pdu_3d_pos_x**[patchIdx]	u(v)
patch_pos_in_view_y [a][p]	u(v)	**pdu_3d_pos_y**[patchIdx]	u(v)
patch_rotation[a][p]	u(3)	**pdu_orientation_index**[patchIdx]	u(v)

Some mismatches between both syntax are noted:

- The number of patches in V-PCC is not sent and the total number of patches is derived by using the patch mode indicating the end of list.
- The atlas dimension in V-PCC does not have minus1 (which makes sense and should be suggested to the V-PCC group).
- Occupancy map flag is not implemented, since V-PCC explicitly transmits the occupancy map separated from the geometry.
- By using the projection id to indicate the camera, the following alteration is needed in the V-PCC text

pdu_projection_id[p] specifies the values of the projection mode and of the index of the normal to the projection plane for the patch with index p of the current atlas tile group. The value of pdu_projection_id[p] shall be in range of 0 to MaxNumProjectionID, inclusive, where the variable MaxNumProjectionID is derived in the following manner:

MaxNumProjectionID = vps_miv_sps_present_flag ? num_views_minus1 : (asps_extended_projection_enabled_flag ? 17 : 5).

The number of bits used to represent pdu_projection_id[p] is vps_miv_sps_present_flag ? ceil(log2(num_views_minus1)) : (asps_extended_projection_enabled_flag ? 5 : 3)

NOTE: Bart Kroon detected a bug in the DIS syntax, which is currently being fixed. The syntax for intra patches will be revised.

- Entity ID is currently not implemented, but could be sent as an SEI message?
- V-PCC encodes the size of the patches by sending the difference between the current patch and the immediately previous one, this method could be suggested to MIV. Noticed that V-PCC usually sorts the list of patches by size, making this delta encoding more efficient.
- In V-PCC, the position in the atlas is given in units of PatchPackingBlockSize
- For the position in view, V-PCC can use the position in 3D space. If one considers the V-PCC

patch camera frustum as a bounding box with size $2^{gi\_geometry\_3d\_coordinates\_bitdepth\_minus1 + 1}$, then the position in 3D is also equivalent to the position in view *(see discussion on patch representation as a volumetric slice of the capturing camera in Section 1)*.

- The patch orientation is exactly the same, only the MIV extension would enforce the number of orientations to be equal to 8, which is signalled in V-PCC by setting asps_use_eight_orientations_flag equal to 1.

3 V-PCC branch

In this section we provide a brief description of the branch of the V-PCC reference software [7] that was modified to encode MIV content.

The software is a full integration of the V-PCC reference software (TMC2v8.0 [9]) and the MIV reference software (TMv3.0 [8]). The extension of the input file is used to select the coding routine that generates the patches and creates the V-PCC compliant bitstream. If ply is used, the software assumes that we are coding a point cloud, and calls the encoding function from the original TMC2v8.0, which will generate patches from a point cloud. However, if a json file is used, then the software assumes that the content to be coded is a MIV content, that is, multiple views and multiple depth maps described by a json file. The original TMv3.0 coding routine is called to generate the patches, the mapping between the MIV patches and the V-PCC patches is done, and the content is encapsulated using the V-PCC specification. Figure 5 shows a typical V-PCC bitstream carrying point cloud content, and Figure 6 depicts the same V-PCC software carrying MIV content.

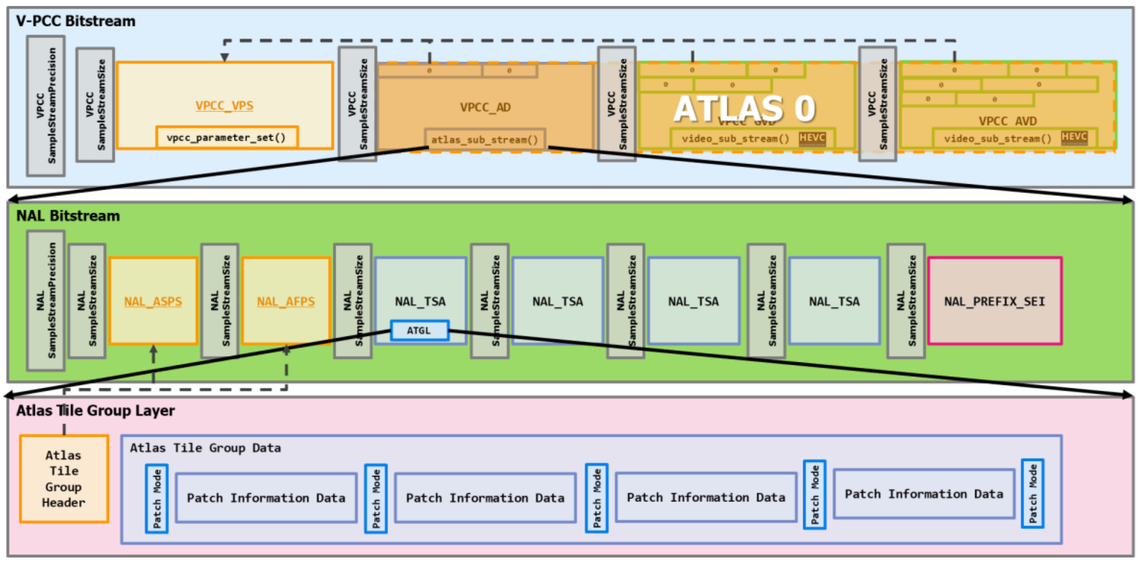

Figure 5: V-PCC bitstream for point cloud coding

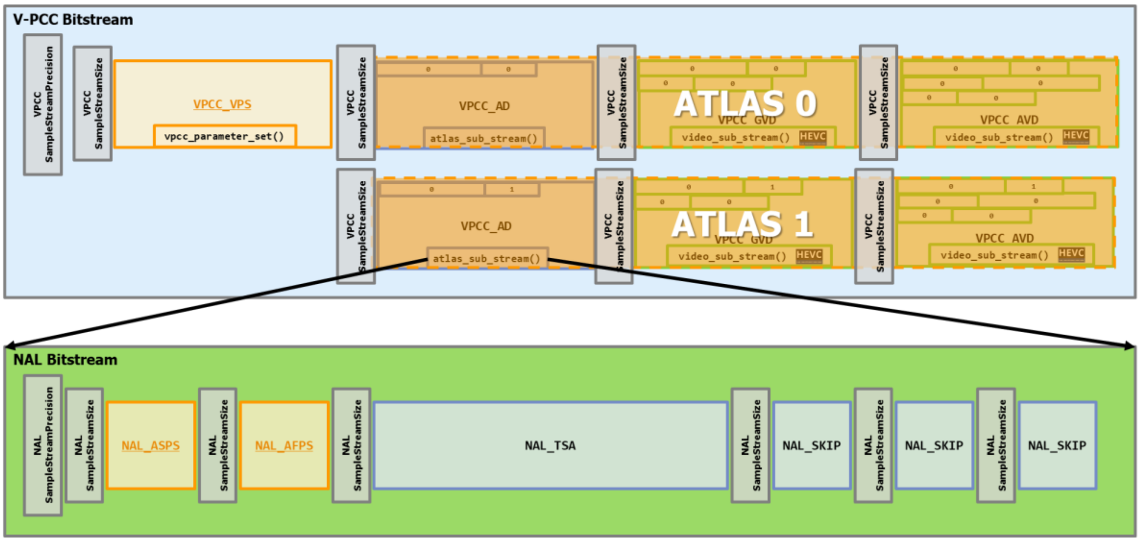

Figure 6: V-PCC bitstream for MIV content coding

Notice that even though the encoder has different patch generation functions, the decoder is exactly the same for both content. Therefore, the decoder generates a point cloud representation of the MIV content by using the camera model described in the MIV sequence parameter set. The software calculates a homography transform of the patch data to the 3D world space, and reconstructs the point cloud. Looking the reconstructed point cloud from a different angle one can see that the input has higher quality for viewing positions similar to the capturing camera, but poor quality for viewing positions far away from the capturing camera position. One can see several isolated points and noise along the surface of objects when the viewing position changes, as shown in Figure 7. A post-processing technique can take into account the capturing camera position for better representation of the point cloud. Furthermore, a better patch generation considering the point cloud characteristics can improve the reconstruction as well.

Figure 7: Reconstructed point cloud for MIV sequences, (left) from a favorable viewing position, (right) from a viewing position distant from the original capturing camera position (top position)

4 Simulation results for MIV sequences

In this section we provide partial results for the modified V-PCC software [7] for V-PCC CTC results [5] and MIV CTC results [6].

For the V-PCC dataset, a full check of all sequences as described in the CTC [5] was performed. Table 2 and Table 3 below show that the current software achieves the exact same result as the V-PCC reference software. This means that when point cloud content is being encoded, the modification to accommodate MIV content does not affect the current standard, as expected.

Table 2: Comparison between V-PCC reference software anchor results and the modified V-PCC code for MIV sequences (32 frames)

Reference:	Sony Cross-check TMC2v8.0 (12/13/2019)
Tested:	MIV code base (CTC 32 frames)

All Intra

C2_ai	lossy geometry, lossy attributes [all intra]									
	Geom. BD-TotGeomRate [%]		End-to-End BD-AttrRate [%]			Geom. BD-TotalRate [%]		End-to-End BD-TotalRate [%]		
	D1	D2	Luma	Chroma Cb	Chroma Cr	D1	D2	Luma	Chroma Cb	Chroma Cr
Cat2-A average	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%
Cat2-B average	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%
Cat2-C average	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%
Overall average	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%
Avg. Enc Time [%]	100%									
Avg. Dec Time [%]	104%									

CW_ai	lossless geometry, lossless attributes [all intra] bpip ratio [%]		
	Tot.Geom	Colour	Total
Cat2-A average	100.0%	100.0%	100.0%
Cat2-B average	100.0%	100.0%	100.0%
Cat2-C average	100.0%	100.0%	100.0%
Overall average	100.0%	100.0%	100.0%
Avg. Enc Time [%]		99%	
Avg. Dec Time [%]		116%	

Inter, Low Delay

CW_ld	lossless geometry, lossless attributes [inter, low delay] bpip ratio [%]		
	Tot.Geom	Colour	Total
Cat2-A average	100.0%	100.0%	100.0%
Cat2-B average	100.0%	100.0%	100.0%
Cat2-C average	100.0%	100.0%	100.0%
Overall average	100.0%	100.0%	100.0%
Avg. Enc Time [%]		90%	
Avg. Dec Time [%]		116%	

Inter, Random Access

C2_ra	lossy geometry, lossy attributes [inter, random access]									
	Geom. BD-TotGeomRate [%]		End-to-End BD-AttrRate [%]			Geom. BD-TotalRate [%]		End-to-End BD-TotalRate [%]		
	D1	D2	Luma	Chroma Cb	Chroma Cr	D1	D2	Luma	Chroma Cb	Chroma Cr
Cat2-A average	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%
Cat2-B average	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%
Cat2-C average	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%
Overall average	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%
Avg. Enc Time [%]	100%									
Avg. Dec Time [%]	107%									

Table 3: Comparison between V-PCC reference software anchor results and the modified V-PCC code for MIV sequences (all frames)

Reference:	ALL_master+r8.0
Tested:	MIV Code base (Jan 1st)

All Intra

C2_ai	lossy geometry, lossy attributes [all intra]									
	Geom. BD-TotGeomRate [%]		End-to-End BD-AttrRate [%]			Geom. BD-TotalRate [%]		End-to-End BD-TotalRate [%]		
	D1	D2	Luma	Chroma Cb	Chroma Cr	D1	D2	Luma	Chroma Cb	Chroma Cr
Cat2-A average	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%
Cat2-B average	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%
Avg. Enc Time [%]	116%									
Avg. Dec Time [%]	125%									

CW_ai	lossless geometry, lossless attributes [all intra] bpip ratio [%]		
	Tot.Geom	Colour	Total
Cat2-A average	100.0%	100.0%	100.0%
Cat2-B average	100.0%	100.0%	100.0%
Avg. Enc Time [%]		125%	
Avg. Dec Time [%]		120%	

Inter, Low Delay

CW_ld	lossless geometry, lossless attributes [inter, low delay] bpip ratio [%]		
	Tot.Geom	Colour	Total
Cat2-A average	100.0%	100.0%	100.0%
Cat2-B average	100.0%	100.0%	100.0%
Avg. Enc Time [%]		122%	
Avg. Dec Time [%]		118%	

Inter, Random Access

C2_ra	lossy geometry, lossy attributes [inter, random access]									
	Geom. BD-TotGeomRate [%]		End-to-End BD-AttrRate [%]			Geom. BD-TotalRate [%]		End-to-End BD-TotalRate [%]		
	D1	D2	Luma	Chroma Cb	Chroma Cr	D1	D2	Luma	Chroma Cb	Chroma Cr
Cat2-A average	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%
Cat2-B average	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%
Avg. Enc Time [%]	117%									
Avg. Dec Time [%]	124%									

Note: Basketball and Dancer sequences were simulated with different number of frames, therefore they were removed from the comparison. The excel files in attachement to this contribution provides more details into the simulations.

For the MIV dataset, only a limited number of sequences were used, namely: classroom, museum, painter and frog sequences. Since the main interest was to determine the efficiency of metadata coding using the V-PCC specification, results for only the first rate point are reported. Table 4 and Table 5 show the results obtained for 17 and 97 frames, respectively, with the modified V-PCC software. It can be noted that the

Metadata bitstream is very small compared to the geometry and texture data. Using the V-PCC notation, the metadata can be transmitted with less than 100kbps for all tested sequences. More details can be found in the attached excel and log files.

Table 4: MIV results using the proposed software (17 frames)

Class	Sequence	Rate [Mbps]	Bits per input point [Mbps]			
			Total	Geometry	Metadata	Colour
MIV	classroom	117.818	117.82	4.45	0.06	113.31
	museum	58.552	117.818	4.450	0.061	113.307
	painter	76.704	76.70	51.59	0.09	25.02
	frog	408.883	117.818	4.450	0.061	113.307

Table 5: MIV results using the proposed software (97 frames)

Class	Sequence	Rate [Mbps]	Bits per input point [Mbps]			
			Total	Tot.Geom	Geometry	Colour
MIV	classroom	91.636	91.64	5.66	0.05	85.93
	museum	43.102	91.636	5.665	0.045	85.926
	painter	80.173	80.17	58.00	0.08	22.09
	frog	337.121	91.636	5.665	0.045	85.926

5 Conclusion and remaining issues

We have successfully shown that current MIV content can be encoded with the proposed modified V-PCC software, and that the V-PCC syntax can accommodate MIV content. Nevertheless, some parts of the current MIV working draft were not considered, like the viewing space information, the group ID, or the depth occupancy information. We believe that some of these items could be coded as SEI messages, or even included in the proposed MIV extension for V-PCC, but further discussions should happen between the groups for better aligment.

6 Acknowlegment

I would like to thank Bart Kroon for the many useful discussion and for so diligently looking into the V-PCC spec. Your work has greatly improved our specification.

7 References

[1] N18935, "Description of Immersive Video Core Experiments 1: Metadata", MPEG 128, October 2019

[2] M51044, "[V-PCC][specification] On the integration between MIV and V-PCC", MPEG 128, October 2019

[3] N18479, "Continuous Improvement of Study Text of ISO/IEC CD 23090-5 Video-based Point Cloud Compression", MPEG 126, March 2019

[4] N18794, "Working Draft 3 of Metadata for Immersive Video", MPEG 128, October 2019
[5] N18883, "Common Test Conditions for PCC", MPEG 128, October 2019
[6] N18789, "Common Test Conditions for Immersive Video", MPEG 128, October 2019
[7] https://gitlab.com/mpeg-i-visual/vpcc_miv_m51044
[8] http://mpegx.int-evry.fr/software/MPEG/MIV/RS/TM1.git
[9] http://mpegx.int-evry.fr/software/MPEG/PCC/TM/mpeg-pcc-tmc2.git