**3GPP TSG-SA SA4#131-bis-eS4-250448**

**Online, 11 – 17 April 2025**

|  |
| --- |
| *CR-Form-v12.2* |
| **CHANGE REQUEST** |
|  |
|  |  | **CR** |  | **rev** | **2** | **Current version:** | **0.3.0** |  |
|  |
| *For* [***HE******LP***](http://www.3gpp.org/3G_Specs/CRs.htm#_blank)*on using this form: comprehensive instructions can be found at* [*http://www.3gpp.org/Change-Requests*](http://www.3gpp.org/Change-Requests)*.* |
|  |

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| ***Proposed change affects:*** | UICC apps |  | ME |  | Radio Access Network |  | Core Network |  |

|  |
| --- |
|  |
| ***Title:***  | pCR on update of information for dense dynamic point cloud representation format |
|  |  |
| ***Source to WG:*** | InterDigital, Sony, Samsung, Nokia, Philips, Deutsche Telekom, Fraunhofer HHI, KDDI, Sony Group Corporation, Huawei |
| ***Source to TSG:*** |  |
|  |  |
| ***Work item code:*** |  |  | ***Date:*** | 2025-06-04 |
|  |  |  |  |  |
| ***Category:*** | B |  | ***Release:*** | Rel-19 |
|  | *Use one of the following categories:****F*** *(correction)****A*** *(mirror corresponding to a change in an earlier release)****B*** *(addition of feature),* ***C*** *(functional modification of feature)****D*** *(editorial modification)*Detailed explanations of the above categories canbe found in 3GPP [TR 21.900](http://www.3gpp.org/ftp/Specs/html-info/21900.htm). | *Use one of the following releases:Rel-8 (Release 8)Rel-9 (Release 9)Rel-10 (Release 10)Rel-11 (Release 11)…Rel-16 (Release 16)Rel-17 (Release 17)Rel-18 (Release 18)Rel-19 (Release 19)* |
|  |  |
| ***Reason for change:*** | In alignment with information agreed for the dynamic mesh format this contribution includes public information from MPEG for use cases of the dense dynamic point cloud representation format |
|  |  |
| ***Summary of change:*** | Updates related to dynamic point cloud representation format  |
|  |  |
| ***Consequences if not approved:*** | Level of information agreed for dynamic mesh and dense dynamic point cloud would be different |
|  |  |
| ***Clauses affected:*** |  |
|  |  |
|  | **Y** | **N** |  |  |
| ***Other specs*** |  |  |  Other core specifications  | TS/TR ... CR ...  |
| ***affected:*** |  |  |  Test specifications | TS/TR ... CR ...  |
| ***(show related CRs)*** |  |  |  O&M Specifications | TS/TR ... CR ...  |
|  |  |
| ***Other comments:*** | Changes are tracked starting from TR V0.3.0 |
|  |  |
| ***This CR's revision history:*** |  |

**== CHANGE 1 ===**

…

### 4.3.3 Dense Dynamic Point Cloud representation format

There are many applications for point clouds such as representing highly accurate maps of landscapes, buildings, infrastructure, etc… but the format is also used to represent people, animals, objects and scenes composed from these. More precisely, for representing people and objects dense dynamic point clouds are in focus.

#### 4.3.3.1 Definition

A point cloud frame is defined as set of (x,y,z) coordinates, where x,y,z have finite precision and dynamic range, ,depending on the data type that is used for representing the coordinates. Each (x,y,z) can have multiple attributes associated to it (a1 ,a2, a3 …), where the attributes may correspond to color, reflectance, transparency, normals or other properties of the object/scene that would be associated with a point. Colour is typically represented as RGB and a normal is a normal to a point which can be used by the renderer for handling lighting. Typically, each point in a point cloud frame has the same number of attributes attached to it. Dynamic point clouds consist of several consecutive point cloud frames with the same coordinate system, precisions and attributes. The number of points typically changes from one frame to the other and there is no relation between a point of one frame to the other frame. A dense point cloud contains a high density of points with close neighbors (typically more than 500.000 points per frame for a person or object), where a renderer is able to produce a closed surface allowing for a highly detailed representation.

A simple and often used file format for point clouds is the Polygon File Format (PLY) that has been developed by Greg Turk at Stanford University in 1994 [D1]. Other formats, like the Object File Format (OBJ) can also be used to represent point clouds.

MPEG has defined several use cases for point cloud compression, including *Real-time 3D Immersive Telepresence*, *Content AR/VR viewing with Interactive Parallax* and *3D Free viewpoint Sport Replays Broadcasting* [VOL-XX]. The typical characteristics of the point clouds in these use cases are summarized in Table. 4.3.3.1-1

Table 4.3.3.1-1 Typical Characteristics of Point Clouds in MPEG-Defined Use Cases

|  |  |  |  |
| --- | --- | --- | --- |
| **Use Case** | **Number of Points**  | **Color Representation** | **Additional Properties** |
| **Real-time 3D Immersive Telepresence** | To represent a reconstructed human: Between 100,000 and 10,000,000 points per frame | 8-10 bits per color component | Normals and/or material properties for shader rendering |
| **Content AR/VR Viewing with Interactive Parallax** | To represent closeby objects in the scene: Between 100,000 and 10,000,000 points per frame | 8-10 bits per color component | Global parameters defining the spatial constraints of the rendering viewport |
| **3D Free Viewpoint Sport Replays Broadcasting** | 100,000 – 100,000,000 points per frame | 8-12 bits per color component | Can contain multiple clusters/groups of points (different players) |

Editor’s Note: Will be completed during study

#### 4.3.3.2 Production and Capturing Systems

Professional capturing of volumetric video is typically done with a rig of synchronized cameras aligned around the asset(s) to be captured. Depending on the rig, there can be one or more layers of cameras at different height positions, with each layer consisting of up to 60 cameras. Cameras can be equipped with depth sensors. Hardware such as cameras and depth sensors are typically off the shelf equipment, but the assembly in the rig is vendor dependent and proprietary.

The various camera and depth sensor signals are fed into a production pipeline that produces the asset. Production includes stitching the various signals, filling holes, correcting occlusions, etc. Persons or physical objects (e.g., a ball or an instrument) can be combined in an asset or separate assets can be used for each person or object. The representation format of a produced asset is typically a dense dynamic point cloud or a dynamic mesh.

The Volumetric Format Association (VFA) [D2] aims to “Drive the development of volumetric video as the next revolution for content creation, editing 3D content, distribution of 3D content and creating entirely new ways to tell stories and communicate with each other”. One result of their work is an end-to-end workflow consisting of Volumetric Capturing, Volumetric Processing, Volumetric Encoding and Decode/Render. The workflow can be downloaded from their website in [PDF](https://www.volumetricformat.org/_files/ugd/f2416f_3e1aeca4db234afcae9a8c15ea4f610a.pdf) (https://www.volumetricformat.org/\_files/ugd/f2416f\_3e1aeca4db234afcae9a8c15ea4f610a.pdf) format. Volumetric Capturing is in line with our description above. Volumetric Processing shows the dynamic point cloud representation format as a central element. First a raw point cloud is created, and which is further processed (e.g. fill holes) and converted to the produced asset. Representation formats for the produced assets is either a dynamic point cloud (in the workflow named as a patch-based format) or a dynamic mesh.

The Volumetric Encoding step includes both options, point cloud and mesh. Once streamed and received on a device, the Decode/Render step includes rendering the mesh, the point-cloud as is or generating mesh or voxels prior to rendering.

#### 4.3.3.3 Rendering and Display Systems

The dense dynamic point cloud representation format can be rendered to 2D displays such as in mobile phones, tablets, TV sets but also to HMDs or other 3D type displays.

The visual viewing quality of the point cloud format depends heavily on how voxels are rendered. Just reconstructing voxels in 3D space may bring a limited viewing experience and holes/cracks may become visible. To show the impact of rendering two renderers are investigated:

- MPEG renderer: Each voxel is replaced by a cube of a configurable fixed size. This renderer is deliberately simple for studying the pure impact of compression.

- Representative renderer: Each voxel is replaced by a splat of a size that depends on the viewing distance and some blending is implemented to avoid flickering of points. There are no sophisticated techniques such as lighting or use of normals integrated. It represents a minimum of what a device manufacturer would do to prevent holes or cracks to preserve a good subjective experience. It is not state-of-the-art or most sophisticated renderer possible.

In the following we give an example of the impact of the renderer on the head of the sequence Thomas with Vox 10 conversion:



Figure 4.3.3.3-1 Vox 10 MPEG renderer Figure 4.3.3.3-2 Vox 10 Representative renderer

Both snapshots are rendered from the same Vox 10 sequence. In Figure 4.3.3.3-1 we see far more cracks and holes and the borderline of the sequence is less smooth. However, the eyebrows look a bit sharper in Figure 4.3.3.3-2. A high-end industry renderer may do better than the renderers illustrated here.

When evaluating or comparing the point cloud representation format it is essential to select a renderer that is representative of a minimum of what the industry would implement, as holes and cracks in images would influence evaluations negatively.

More sophisticated renderers in products could fill better potential holes, recreate detail and apply lighting depending on the scene. The point cloud representation format supports normals which are useful for lighting the scene. When rendering a point cloud sequence in a scene, correct lighting including shadows and colour alignment can greatly impact the realism of the resulting experience.

POINTS\_GL is the simplest OpenGL[D3] primitive type used for rendering (lines and polygons are others that are also commonly used) and a point cloud can be interpreted as a vertex stream that represents points (after ordering of the points). Therefore, a point cloud can be rendered in an extremely straightforward way using native OpenGL vertex shaders. The supported rendering in the standard OpenGL specified by the Khronos consortium implies that point clouds can be rendered on devices that support OpenGL which is rather common today. OpenGL vertex shader renders points size larger than zero, this can be set GL\_PROGRAM\_POINT\_SIZE as a configuration of the rendering.

Specific optimizations for rendering are device manufacturer dependent.

#### 4.3.3.4 Support Information

##### 4.3.3.4.1 Test and reference sequences

<TBD>

##### 4.3.3.4.2 Uncompressed data size

The uncompressed data size of a point cloud frame depends on the number of points and the number of attributes. The following table gives data size examples and raw bitrates for the sequence Thomas.

Table 4.3.3.4.2-1 Uncompressed data size and bitrate

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Sequence | Quantization | #frames | #points | mean frame size (bytes) | bitrate (mbps) |
| Thomas | Vox10 | 32 | 19012250 | 4010396 | 979.10 |
| Thomas | Vox11 | 32 | 76336020 | 16996692 | 4149.58 |
| Thomas | Vox12 | 32 | 305897397 | 71694702 | 17503.59 |

##### 4.3.3.4.3 Known compression technology

Visual volumetric video-based coding (V3C) and video-based point cloud compression (V-PCC) [D4]

Geometry-based point cloud compression (G-PCC) [D5]

Draco [D6]

##### 4.3.3.4.4 Conversion from other formats

Point clouds can be obtained by sampling from surface-based formats such as meshes. Such transformation is lossy. There are different sampling methods (e.g. methods based on face sampling, on texture map sampling, on ray casting from a grid, etc.) and it’s up to the content provider to select the appropriate sampling method depending on the content and creative intent.

##### 4.3.3.4.5 Typical quality criteria

The visual quality of a point cloud depends on the number of points (density) in the point cloud. For attributes colour is mandatory and there may be reflectance, transparency and normal. Colour is typically in RGB with each in 8 bits. Reflectance, transparency and normal can be used by the renderer when the point cloud is rendered in a scene.

Point clouds of around 1M points/frame allow to watch from a wider distance (e.g. from 3m\*) and 2M points/frame allow to get closer (e.g to around 1.5m distance) at good quality for the target scenario. Emotional facial expressions and buttons and tissue structure of cloths is visible. More points per frame improve the details, but this may not be required for the target scenario. But if a scenario would require it, a professional volumetric video production system is able to capture details from e.g. skin or finer details of tissue and it can be represented with the point cloud representation format.

\* A typical demonstration scenario would be to use e.g., a smartphone or tablet running a volumetric video application showing a real person of e.g., 3m distance on the screen captured by the camera and rendering at the same time a second person rendered from a point cloud next to the first person.

Other scenarios may require the representation of the full detail of a person and the number of required points can be approximated as follows:

Assumptions:

- The visual resolution of the human eye is 1/60 of a degree

- Average human body surface is about 1.9m2

- For simplification the body surface is approximated as a square

Number of points = 1.9/((tan1/60 \* d)^2), where d is the viewing distance from the person.

This leads to the following number of points:

- 1.5m distance: 10 M pixels

- 3m distance: 2.5 M pixels

#### 4.3.3.5 Benefits and Limitations

##### 4.3.3.5.1 Benefits

Point cloud representation is simple in structure and representation, has high accuracy and resolution, is faithful to original data, and is easy to acquire from sensors or cameras. Point cloud generation needs less pre-processing as there is no need for surface reconstruction, if sensor data is not so noisy.

A point cloud can be rendered in an extremely straightforward way using native OpenGL vertex shaders.

##### 4.3.3.5.2 Limitations

Point-cloud data does not include information on surfaces and is harder to edit or transform.

Editor’s Note: Will be completed during study.

**== CHANGE 2 ===**

# 2 References

[VOL-XX] MPEG 115, Use cases for Point Cloud Compression, [https://mpeg.chiariglione.org/sites/default/files/files/standards/parts/docs/w16331\_Use\_Cases\_for\_Point\_Cloud\_Compression\_(PCC)\_0.docx](https://mpeg.chiariglione.org/sites/default/files/files/standards/parts/docs/w16331_Use_Cases_for_Point_Cloud_Compression_%28PCC%29_0.docx)