**Source: Deutsche Telekom, Fraunhofer HHI, Huawei, InterDigital, KDDI, Nokia, Philips, Samsung and Sony**

**Title: [FS\_Beyond2D] On representation format – Dynamic Point Cloud representation format**

**Agenda item: 9.9**

**Document for: Discussion and Agreement**

# Introduction

Contribution S4aV240040 proposes a template for collecting information about relevant Beyond 2D video representation formats. The present contribution reuses this template and provides information for the dynamic point cloud representation format in relation to the scenario described in [1].

# Dynamic Point Cloud representation format

##  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 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 [2]. Other formats, like the Object File Format (OBJ) can also be used to represent point clouds.

##  Production and Capturing Systems

High-quality 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 dynamic point cloud or a dynamic mesh.

The Volumetric Format Association (VFA) [3] 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) 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.

##  Rendering and Display Systems

As other volumetric video formats, the 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.

Chapter 5 of contribution S4-241494 discusses the impact of rendering and illustrations in chapter 7 of the same document show the result of a simple renderer that basically represents a voxel by a cube and a more sophisticated renderer named industry renderer that replaces a voxel by a splat where the size is adapted depending on viewing distance and some blending is implemented to avoid flickering of points. 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 could fill better potential holes, recreate detail and apply lighting depending on the scene. The point cloud representation format supports reflectance and 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 [8] 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.

##  Support Information

**Test and reference sequences**

Document S4aV240021 [5] provided an initial list of candidate sequences including a proposal for a JSON schema. Some of these sequences are under license agreement and must be protected with a password when stored on a server.

**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 sequence Thomas.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 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 |

Table 1 Uncompressed data size and bitrate

**Known compression technology**

Visual volumetric video-based coding (V3C) and video-based point cloud compression (V-PCC) [6]

**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.

**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.

Contribution [4] gives quality examples of the point cloud representation format and we see that 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 scenario described in [1]. But if a scenario would require it, a high-end 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.

##  Benefits and Limitations

**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

**Limitations:**

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

# Conclusion

The submitters propose to include the information collected about the point cloud representation format in the present document in the PD.

# References

[1] InterDigital et al., S4-241197 - Scenario on Streaming of Beyond 2D Produced VoD Content - Use Case “Volumetric Video with single asset”, SA4 128 Jeju – Korea, May 2024

[2] Greg Turk, [The Polygon File Format](http://gamma.cs.unc.edu/POWERPLANT/papers/ply.pdf), Stanford University, 1994

[3] Volumetric Format Association VFA, <https://www.volumetricformat.org/>

[4] InterDigital et al., S4-241494, Quality examples of the point cloud representation format for streaming single asset scenario, SA4 129-e - Online

[5] InterDigital et al., S4aV240021, Candidate sequences for “streaming single asset” scenario, SA-4 128 post, June 2024

[6] V-PCC, Visual volumetric video-based coding (V3C) and video-based point cloud compression (V-PCC), ISO/IEC 23090-5 2nd Ed, Nov 2023