Source: Samsung Electronics Co., Ltd.

**Title: FS\_5GSTAR: Content formats and codecs pCR**

**Agenda Item: 11.9**

**Document for: Discussion and Agreement**

# **Introduction**

The motivation of this contribution is to study the current state of the art immersive media content formats (including possible codecs and their output bitstreams), and to identify those that might be relevant to AR/MR use cases and device architectures as defined in FS\_5GSTAR. Through a mapping between the different content formats and the current functions as defined in the device architectures, we hope to initiate discussions on potential gaps and requirements regarding the different technologies.

In further detail, we propose a list of content formats and codecs for type 1: STAR-based and type-2: EDGAR-based architectures that are defined in clause 4.2.2 of TR 26.998. After a mapping with the device architecture AR/MR functions and their possible media inputs, we draw a conclusion that immersive media, compressed immersive media and compressed 2D media should be considered as input formats for the devices, and in additional also identify the need for other AR/MR related metadata, namely user pose, and camera information.

# **Considerations**

After reviewing this great contribution, we identify that it seems that we have focused on media processing aspects in the device architectures, but missed two main aspects in the both device types:

1. An AR/MR scene renderer/handler/manager/presentation/viewer: This function sets up the scene and its relation to the real world and also manages the presentation of virtual objects in this scene and by this, in the real world. This can be part of the application, but preferably a dedicated function is added. This function also sets up connectivity, decoding and rendering workflows in a dynamic session. The handler arranges individual objects dynamically and puts it into the context of the scene. Preferably, this is described by a graph and we get the notion of nodes in the graph. A graph can be simple and have a single node, but can also be complex with many nodes.
	1. For STAR, the scene handler may have many functionalities and can handle many different nodes
	2. For EDGAR, the scene handler may be simple and most of the processing is in the edge. Maybe it can only handle a single node (1 object)
2. Connectivity engine/Media Access Function: This function accesses media streams on the network or on the local device and provides the decoded media to the scene renderer which maps the decoded media to a node and renders the buffers in an iterative fashion.

We want to make sure that people are not “overloaded” by a specific instantiation of a scene graph/description, we have an abstracted view – more a data model. There are likely different ways to describe graphs, but the concept of nodes is probably quite ubiquitous.

A few scene description formats exist and clause 4.2 documents an overview.

Within the scene, in general and based on the above, each of the nodes gets assigned some properties that relate to the scene (position, space, etc.), but also some associated formats that are understood by the graph to be rendered in the scene dynamically, depending on the users pose.

Nodes have for example associated the following information

* Formats
* Physical instantiation of media (codec, access link, etc.)

In the context of this document,

* several visual media representation formats are documented in clause 4.1
* several compression technologies for some of these formats are documented in clause 4.3
* camera formats are documented in clause 4.X

We believe it is important that for use cases we create call flows. Preferably we should start with a simple use case and work as follows:

* Call flows
	+ Mapping with use cases; from simple one that showing role of scene description, referring stream-able immersive media,
		- Device is registered to the world
		- Select AR media; the dancer (scene + plugged-in media)
		- Register the dancer object on device’s registered world (as if it’s on real environment)
		- Put some accessories on the dancer.

# **AR/MR functions in device architectures**

As defined in 4.2.1, AR/MR functions include encoding, decoding, rendering and compositing of AR/MR content, after which localization and correction is performed based on the user’s pose information.

STAR-based architecture has both basic AR functions and AR/MR functions on the device. EDGAR-based architecture has only basic AR functions on the device.

Since AR/MR functions are on-device for the STAR-based architecture, immersive media including 2D media can be considered as the input media for the architecture.

Examples of immersive media are 2D/3D objects such as overlay graphics and drawing of instructions [A.2 UC16], 3D objects such as furniture, a house and an animated representation of 3D modeled person [A.3 UC17], a photorealistic volumetric video of a person [A.4 UC18], a 3D volumetric representation of conference participants [A.5 UC19], 2D video, and volumetric information and simple textual overlays [A.6 UC20].

For the EDGAR-based architecture, basic AR functions are on-device therefore 2D media and additional information (such as depth map) generated from immersive media renderer can be considered as the input media for basic AR functions. A rasterized and physically-based rendering (PBR) image is an example of 2D media.

The functions listed in clause 4 of TR 26.998 include:

[AR/MR functions]

* Immersive media encoder, immersive media decoder, compositor, immersive media renderer, immersive media reconstruct process, and semantic perception process.

[Basic AR functions]

* 2D media encoder, 2D media decoder, vision engine and pose corrector

According to the input of each AR/MR function, the media format for each function can be summarized as follows:

[AR/MR functions]

* Input of immersive media encoder, immersive media renderer
	+ Immersive media with user pose
* Input of immersive media decoder, **STAR-based architecture**
	+ Compressed immersive media
* Input of compositor, pose corrector
	+ 2D media with user pose information
* Input of reconstruct process
	+ 2D media with camera information
* Input of semantic perception process
	+ 2D media

[Basic AR functions]

* Input of 2D media encoder
	+ 2D media
* Input of 2D media decoder, **EDGAR-based architecture**
	+ Compressed 2D media
* Input of vision engine
	+ Sensory data, 2D media

As specified in bold, not only do functions have inputs, but each device type also has an input. The input to each device type is output from the 5G system, such as from the 5GMS AS.

A study into the existing technologies to be considered as inputs to each function are identified and presented as a non-exclusive list below.

# **Considerations on AR/MR functions and their input media formats**

# **Immersive media**

# **General**

Immersive media is a media which can be used to provide an immersive experience to users. The immersive experience may include a volumetric presentation of such media. The volumetric presentation does not bind to a specific display technology. For example, a mobile phone can be used to present either the whole AR media, or a part of the AR media. Users can see a volumetric presentation of a part of the AR media augmented in real space. Therefore, immersive media includes not only volumetric media formats such as omnidirectional visual formatsERP image, 3D meshesPrimitives, point cloudsPrimitives, light fieldsPlenopotic image, scene description, and 3D audio formats, but also 2D video2D image as studied in TR 26.928. Elements that construct 3D object(Primitive), projection of volumetric scene(2D image) and their composition in volumetric space(Scene description) can be considered as the components of the immersive media.

# **Primitives (Vertex, Edge, Face, Attribute, Texture)**

3D meshes and point clouds consists of thousands and millions of primitives such as vertex, edge, face, attribute and texture.

Primitives are the very basic elements in all volumetric presentation. A vertex is a point in volumetric space, and contains position information in terms of three axes in coordinate system. In a Cartesian coordinate system, X, Y, and Z make the position information for a vertex. A vertex may have one or more attributes. Color and reflectance are typical examples of attributes. An edge is a line between two vertices. A face is a triangle or a rectangle formed by three or four vertices. The area of a face is filled by interpolated color of vertex attributes or from textures.

* File formats for Primitives

OBJ, PLY, and GPU command buffer in OpenGL-based languages (e.g., glTF Buffer) are methods of encapsulating the primitives. A sequence of primitive files – such as multiple OBJs, PLYs or a set of GPU command buffers in a time can present an animation of volumetric presentation.

# **2D media (RGB, Depth, ERP, Plenoptic image)**

2D media can be used to deliver a volumetric presentation. One camera or one view frustum in a scene may return a perspective planar capture of the volumetric scene. Such a 2D capture can consist of pixels with color attributes (RGB).

Each pixel may represent the distance between the surface of a volumetric scene and the camera (or the view frustum). A depth map contains pixels with the distance attribute (Depth). Distance is one-dimensional information and can be represented in an absolute/relative or linear/non-linear manner. Metadata to explain the depth map can be provided.

The capturing of a volumetric scene can also be expressed as an omnidirectional image in a spherical coordinate system. Equirectangular Projection (ERP) is an example projection method to map a spherical coordinate system into a cylindrical coordinate system. The surface of the cylindrical coordinate system can be considered as 2D media.

Capturing of a volumetric scene can be further improved/elevated with hundreds of cameras in an array; HDCA (High Density Camera Array) or lenticular are methods to capture rays of light. Each point surface of a volumetric scene has countless rays of colors in multiple different directions. Each position of a camera captures a different color from the same point surface of the volumetric scene. 2D images from the camera array can be packed together to form a larger plenoptic image.

A sequence of multiple 2D media in time can present an animation of volumetric presentation.

* Formats for 2D media

Still image formats can be used for 2D media. The 2D media may have metadata for each image or for a sequence of images. For example, pose information describes the rendering parameter of one image. The frame rate or timestamp of each image are typically valid for a sequence of such images. The place for metadata encapsulation – whether the 2D media is compressed or uncompressed – can be considered based on the usage of the metadata.

# **Scene description**

A volumetric media containing primitives ranging from one vertex to multiple objects can be described by a scene description. A scene description typically has a tree structure to represent the components of the scene. A primitive or a group of primitives are referenced as a leaf node of the scene tree. A skeleton to allow for motion rigging or an animation of motion of the skeleton in time can present an animation of volumetric presentation.

* Formats for scene description

Khronos glTF2.0 is one example of a scene description format. It has a tree structure and internal/external resource references. Many 3D contents are authored in the glTF format. Since the current version of glTF2.0 does not allow glTF reference other glTF [https://github.com/KhronosGroup/glTF/issues/1831], immersive media formats other than glTF is considered for use of a glTF as scene description.

Apple Universal Scene Description(USD) allows for the layering of USD files and can therefore be considered to have both scene description and transmission format roles at the same time.[https://graphics.pixar.com/usd/docs/Usdz-File-Format-Specification.html]

# **Compressed immersive media formats**

There exist technologies developed to compress each type of immersive media. For 2D image sequence, 2D video codecs can be used. Especially for 2D ERP image sequence, compression of images and relevant metadata information signalling are handled by MPEG OMAF. For compression of volumetric media, MPEG V3C/V-PCC and G-PCC can be considered.

* 2D Video codecs

In general, 2D video codecs can compress the 2D media listed in clause 3.1. However, there are differences in the context of 2D media such as RGB image versus depth map image, 2D image from one planar perspective camera versus ERP, or 2D image from one camera versus HDCA plenoptic image. Such differences can be considered in the proper encoder/decoder coding tools. AVC and HEVC are industry wide examples of 2D video codecs.[ref: TS26.511]

* MPEG OMAF

OMAF consists of two parts; the first part is a pre-processing which includes a projection of spherical volumetric media onto a 2D image, and the second part is an encapsulation of the compressed 2D frame packed image with metadata signalling the projection.

For the compression of the 2D images, 2D video codecs can be considered and the pre-processing operations are agnostic to specific 2D codec technology.

* MPEG V3C and V-PCC

V3C and V-PCC consists of two parts; the first part is a pre-processing which includes the decomposition of a part of the volumetric media into the planar projection of different characteristics, such as texture, geometry and occupancy, and the second part is an encapsulation of the compressed 2D packed images, with metadata signalling the decomposition.

For the compression of the 2D images, 2D video codecs can be considered and the pre-processing operations are agnostic to specific 2D codec technology.

* MPEG G-PCC

G-PCC divides volumetric media into multiple sub-blocks. Triangle (Trisoup) or leaf (Octree) are used as the units of the divisions. A volumetric media is subdivided recursively until no more sub-blocks are left. The dimension (or level) of the tree is relatively large, such as 2^24. Tools including arithmetic encoding are used to encode all the tree information into the bitstream.

# **2D media**

2D media is the output of the immersive media renderer. One view frustum that represents the user’s viewport is placed in a scene, and in turn, a perspective or an orthogonal projection of the volumetric media is produced. To minimise motion sickness, a pose corrector performs a correction of the 2D media at the last stage of presentation. The pose corrector may require additional information such as the estimated or measured user pose that was used for the rendering of the 2D media. For the case that the latest user pose does not match with the estimated user pose, additional information that provides knowledge on the geometry, such as a depth map, can be delivered from immersive media renderer.

As listed in 3.1, 2D image with pixels containing RGB and Depth information can be considered for this purpose. The consideration of 2D media formats described in clause 3.1 can be adopted.

# **Compressed 2D media format**

As discussed in clauses 3.3, 2D image with pixels having RGB and Depth information can be considered for use of the 2D media decoder in the AR/MR device. The same consideration of compressed 2D media formats in clause 3.2 can be adopted.

# **User pose information**

An AR/MR user stands at a position facing a certain direction at a particular moment.

A position can be represented as a geolocation with longitude and latitude. The position can also be represented as a point in a scene. The scene can be represented as a virtual box on a geometry/mesh which represents user’s real environment. When an AR/MR device reports the user position to obtain a split render of the immersive media from a server, the device calculating the user pose should report either a geolocation, a point in a scene or a point in a geometry. Depending on the representation, to make a point in a scene, or a point in a geometry, valid to the immersive media renderer, the server should be aware of the underlying scene or the geometry. A device should update whenever there is any change in the scene or the geometry through user interaction (e.g., rotating a scene by hand gesture) and/or SLAM (e.g., finer modelling of surrounding environment).

A direction can be represented with a rotation matrix, or roll, pitch, and yaw. The direction is relative to a scene/geometry and the scene/geometry has an origin and default direction of the three axes.

The devices representing a user’s pose moves continuously, and if the device is worn on the user’s head, it can be assumed that he or she frequently turns their head around. A set of position and direction information is only meaningful at a certain moment in time. Since the device reports the user pose at around a frequency of 1KHz, any pose information should include a timestamp to specify when it was measured or created. A pose corrector (e.g., ATW and LSR) in a server may estimate the user’s future pose, whilst a pose corrector in a device may correct the received rendered image to fit the latest user pose.

* Formats for user pose

A position in Cartesian coordinate system can be represented by either X, Y and Z or by a translation matrix. A direction can be represented by a rotation matrix or by quaternions.

OpenXR [https://www.khronos.org/registry/OpenXR/specs/1.0/html/xrspec.html] describes a possible format for user pose in clause 2.16. It consists of 4 quaternions for orientation and 3 vectors for position. Timestamp is represented by a 64 bit monotonically increasing nano-second-based integer.

# **Camera information**

Immersive media can be captured by camera(s). The camera parameters such as focal length, principal points, calibration parameters and the pose of the camera all contribute in understanding the relevance between points in the volumetric scene and pixels in the captured image. Photogrammetry is the technology used to construct immersive media from a continuous capturing of images. Depth sensor-based cameras can be used to capture immersive media from one capturing of the volumetric scene

* Formats for camera information

Camera intrinsic parameters can be represented by a camera matrix. Extrinsic parameters can be represented by a transform matrix.

# **Proposal**

We propose to include the text in clause 3 of this to clause 6.2.5 of TR 26.998 as a baseline for media format discussions.