IBC2023

EFFICIENT DELIVERY AND RENDERING ON CLIENT DEVICES VIA MPEG-I STANDARDS FOR EMERGING VOLUMETRIC VIDEO EXPERIENCES

C. Guede[1], P. Fontaine[1], J. Mulard[1], B. Leroy[1], C. Quinquis[1], R. Gendrot[1], S. Gudumasu[2], V. Allié[1], B. Kroon[3], B. Sonneveldt[3], R. Schimanofsky[3]

[1] InterDigital, France, [2] InterDigital, Montreal and [3] Philips, Eindhoven

ABSTRACT

This paper presents a real-time implementation of a platform jointly developed by InterDigital and Philips that showcases use cases leveraging the MPEG volumetric (MPEG-I V3C) and 2D video standards (VVC, HEVC). We will detail how our platform enables interoperability within existing and emerging extended reality (XR) ecosystems, including the acquisition, streaming, and real-time interactive playback of volumetric video on current and future client devices, for use in applications like telelearning, free-viewpoint sport replays, and 3D telepresence in connected ecosystems like the metaverse.

MPEG's Visual Volumetric Video-based Coding (V3C) standard is an extensive framework for the coding of volumetric video, from dynamic point clouds (V-PCC) to multi-view plus depth and multi-plane image representations (MIV), to offer a single bitstream structure with a uniform bridge to systems-level standards. The V3C carriage standard defines how volumetric content can be stored, transported, and delivered to the end-user, and it repurposes existing 2D video hardware decoding capabilities and GPUs to decode and render volumetric video.

INTRODUCTION

The increasing popularity of XR applications is driving the media industry to explore the creation and delivery of new immersive experiences, while pushing engineers and inventors to address the challenges of real video content manipulation.

A volumetric video is comprised of a sequence of frames, and each frame is a static 3D representation of a real-world object or scene capture at a different point in time. Volumetric video is bandwidth-heavy content that can be presented as dynamic point clouds, multi-view plus depth, or multi-plane image representations. These high bandwidth constraints can be reduced through dedicated compression schemes adapted to these types of contents to reach data rates and files sizes that are economically viable in the industry. Standards play a crucial role in ensuring interoperability across these different types of contents and experiences, and this paper presents the Moving Picture Experts Group (MPEG) Visual Volumetric Video-based Coding (V3C) standard [1] as an open standard solution for efficient compression and streaming of volumetric video. The MPEG community has described use cases for the V3C codec [6] [17].

Looking at trends towards metaverse developments, some could consider a transitioning path from current 2D experiences to future metaverse worlds. The presented platform has been developed as such to allow a remote user to access 2D content first and then further engage with the proposed topic with volumetric viewing; allowing the enriched experience to be brought seamlessly to the user.

This paper presents a real-time implementation of a platform jointly developed by InterDigital and Philips. This platform, illustrated in Figure 1Figure 1, ingests pre-recorded 2D and volumetric video content, provides real-time streaming and rendering. Final content is proposed to the user on various devices such as 2D screens, smartphones or tablets, and VR/AR head mounted displays.

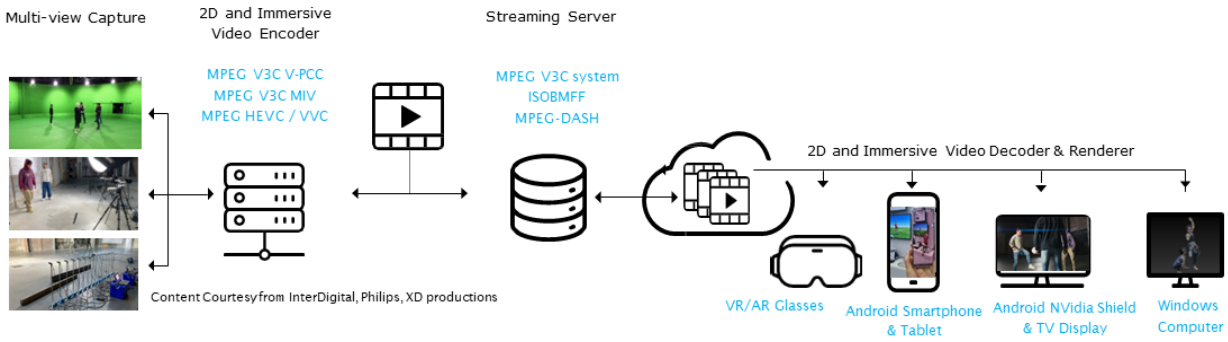

Figure 1 – Immersive Video Decoder Platform - end-to-end view

The organization of the paper is as follows: the standards section will give an overview of the two implemented V3C-based volumetric codecs, namely V-PCC and MIV; along with the V3C carriage layer for systems. The platform architecture section will depict presented Immersive Video Decoder Platform and will highlight principal software components allowing real-time streaming and rendering and proposed integration into XR ecosystem. The evaluation section will provide metrics of the current implementation measured on laptop, smartphones, and tablet devices.

V3C STANDARDS

MPEG has developed two standards that adopt a similar video-based coding approach leveraging traditional 2D codecs (such as HEVC and VVC) but are targeting different volumetric representations and applications, dynamic point clouds and multi-view videos. The common aspects including bitstream structure have been regrouped into the *Visual volumetric video-based coding* (V3C) standard [1]. To enable storage and delivery of such compressed volumetric content, MPEG has developed *Carriage of V3C data* [3], and to composite multiple assets into a single scene, *Scene Description* [4] was created.

Visual Volumetric Video-based Coding (V3C)

V3C [1] maps volumetric data onto one or more flat video frames composed of image patches that can be compressed using any legacy 2D video codec (see Figure 2).

The bitstream structure consists of a sequence of V3C units. The first one is a V3C parameter set (VPS) that provides sufficient information for a decoder to determine if it can handle the bitstream. The remaining V3C units carry atlas and video sub-bitstreams.

Mapping onto flat video frames allows for an efficient decoder/renderer model whereby most of the decoding is offloaded to hardware video decoders and rendering is performed on a GPU. Because the atlases may have multiple video components, decoding a V3C bitstream may require many video decoder instantiations.

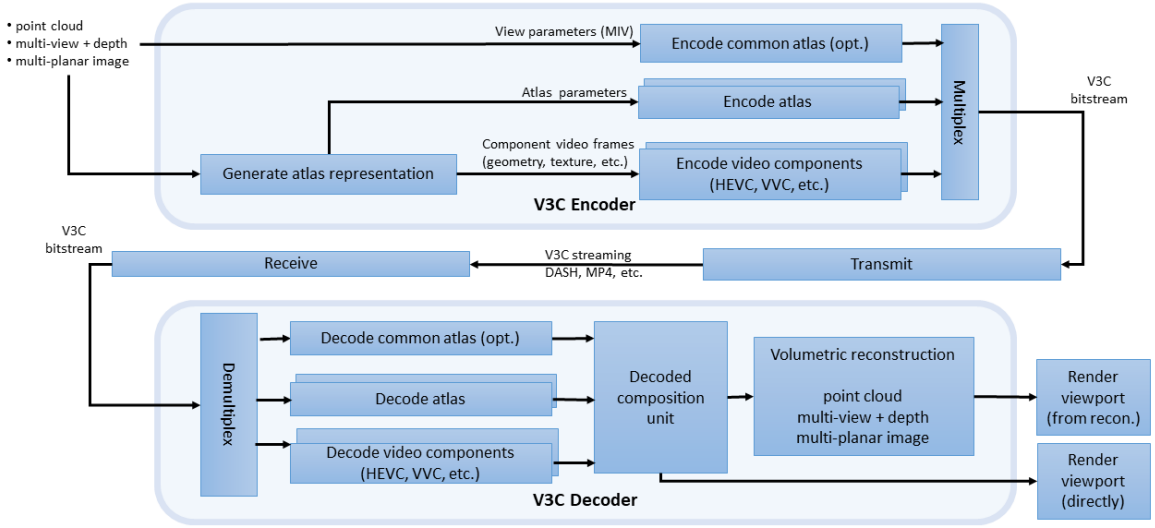

Figure 2 – V3C codec model

Video-based Point Cloud Coding (V-PCC)

V-PCC [1], Annex H handles the dynamic category of Point Clouds identified by the MPEG community, which corresponds to dense point cloud datasets varying in time for applications that have been identified in [16].

The next section provides details on the different encoding and decoding steps shown in Figure 2. Focus on the lossy mode random access configuration is done in the following document even if V-PCC can encode losslessly. Further details on different coding tools and strategy are given in [5].

First, during the atlas generation step, each point cloud frame is projected onto a given number of 2D planes using a cube of orthographic projections by regrouping points into connected components sharing neighbourhood with similar normal. These projections are called patches. Then, patches are arranged into 2D video frames, composed of a trio of component video frames: a 2D geometry frame, that stores the depth values for all connected components, a 2D attribute frame, that stores the corresponding attributes components (e.g., the colour), and a 2D occupancy frame, that indicates which part of the 2D atlas is valid for the 3D reconstruction. The occupancy frame is lossless encoded and subsampled to avoid a high cost in coding. These steps can be done several times as several points may be projected to the same 2D pixel, and several maps could be generated.

In addition, from the atlas generation, associated data are generated to define how to get back to the 3D volumetric frame using 2D representations. They are known as atlas data and represent a relatively small amount of data compared to video sequences. Atlas data also specify encoding tools (see [5]) that are used during the reconstruction to improve the quality of the point cloud.

A performance analysis done in the SMPTE Motion Imaging Journal [8] provides a comparative study on the combination of different V-PCC tools and gives conclusions in terms of objective and subjective tests while comparing different profiles.

By mirroring effect, the V-PCC decoder is a set of three 2D video decoders. These three video components, plus a light parsing of metadata allows the reconstruction of the volumetric data using reconstruction tools defined at the encoder side to improve the point cloud quality. At the end, a set of 3D points to be rendered on the targeted devices is obtained.

MPEG Immersive Video (MIV)

MIV [2] [18] is a set of extensions and profile restrictions on V3C. It supports three image-based volumetric video representations:

- Multi-view + depth (MVD) with texture and geometry video
- Multi-view with only texture video and decoder-side depth estimation
- Multi-planar images (MPI) with texture and transparency video

The main addition that MIV brings to V3C is the transmission of view parameters via V3C common atlas data (Figure 2) comprising camera intrinsics, camera extrinsics, projection type and depth quantization parameters. By transmitting (patches of) multiple cameras it is possible to preserve view-dependent appearance of objects. It also allows for low complexity real-time encoding by postponing part of the 6DoF scene understanding to the client side.

Other extensions are the ability to embed occupancy information in the geometry video, and to associate an entity ID to each patch, thus allowing for object-based coding.

Because MIV is based on one or more views, whether captured from physical cameras or rendered on the encoder side, it is not generally possible to render the volume for any angle because there may be missing data. It would require having cameras all around the scene which is not always feasible or useful. The *viewing space* that is implicitly known and optionally transmitted indicates for which positions and viewing directions a reasonable rendering quality is to be expected. A client may use this information to adjust the rendering or redirect the viewer.

Carriage of V3C Data

The system layer of the V3C standard is defined in the *Carriage of V3C Data* standard [3], which specifies how the V3C data may be carried for different applications.

The carriage standard translates the V3C units in a V3C bitstream to boxes in an ISO base media file format (ISO BMFF) container. This enables existing systems-level standards such as the MPEG-4 file format, which is derived from ISOBMFF, and MPEG dynamic adaptive streaming (DASH) to handle V3C data. The standard introduces three modes for storing V3C-coded content in ISOBMFF: single-track storage, multi-track storage, and non-timed storage. The multi-track encapsulation mode stores the V3C bitstream in the ISOBMFF file in several tracks, where each track represents either part of, or a complete V3C component. This is the preferred mode for streaming applications since independent encoders can run in parallel and the resulting bitstreams can be stored into an ISOBMFF-compliant file, or set of files, as separate tracks. This provides a flexibility where the extraction and direct

processing of each V3C component by their respective decoder becomes much easier without the need to reconstruct the V3C bitstream. The V3C carriage standard also defines how to signal V3C content in the Media Presentation Description (MPD) for DASH-based delivery for both the single-track and multi-track encapsulation modes. This includes defining V3C-specific DASH descriptors as well as restrictions on the DASH segments generated for the V3C content. As with encapsulation, the multi-track mode provides more flexibility by enabling adaptation across several dimensions as each V3C component is represented by its own Adaptation Set. A streaming client can then prioritize or completely drop some components or maps when making adaptation decisions. V3C video component representations can be encoded using different video codecs or different bitrates to allow for efficient adaptive bitrate streaming.

IMMERSIVE VIDEO DECODER PLATFORM ARCHITECTURE

The objective of the Immersive Video Decoder Platform (Figure 3) is to enable real-time rendering of 2D and volumetric V3C video content to allow smooth integration of these technologies into XR applications. A streaming server processes 2D and V3C bitstreams into DASH segments which can then be streamed to multiple video clients over Internet networks. On the client side, the Immersive Video Decoder Platform receives DASH segments via the DASH client and passes data chunks to the decoder which is connected to a host application interfacing with the end-user.

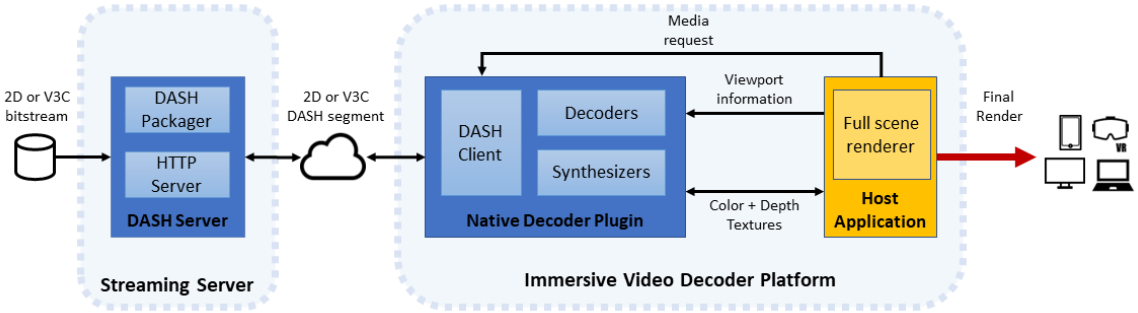

Figure 3 – Immersive Video Decoder Platform global architecture

Details of the Immersive Video Decoder Platform implementation are presented below, where focus is put on V-PCC and MIV content but 2D content will follow the same architecture path.

Native Decoder Plugin

The native decoder plugin (C++, OpenGL) manages both the decoding and rendering of a V3C bitstream. It implements the Unity native render plugin API to leverage the engine cross-platform capabilities and the numerous XR devices supported. It is composed of 5 main stages: a data interface stage, a demultiplexing stage, a decoding stage, a scheduling stage, and a synthesizing stage, as shown in Figure 4.

For the data stage, the bitstream data is fetched either remotely using DASH or locally and split into timestamped chunks delivered to the next stage. The demultiplexing stage uses the parsing tools provided with the relevant test models (either Test Model for Category 2 R22.0 [9] or Test Model for MPEG Immersive Video 14 [10]) to extract the video bitstreams and atlas data from the V3C bitstream units.

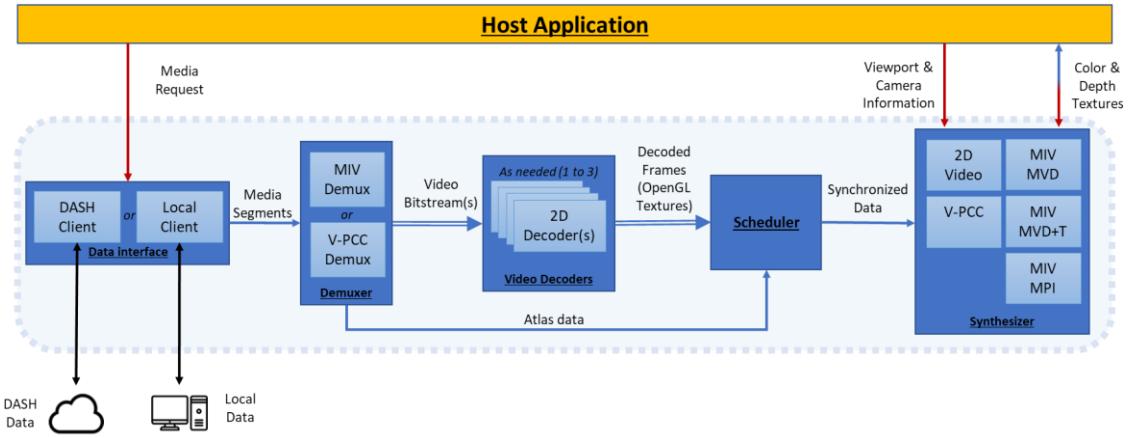

Figure 4 – Native decoder plugin High Level Architecture

V3C contents use multiple video streams, and the number of videos to be decoded varies. This stage is responsible for starting the right number of decoders: 1 video decoder for 2D content, 3 for V-PCC, 2 for MIV MVD or MIV MPI, 3 for MIV MVD+T; and providing them with data. The atlas data skips the decoding stage and is directly sent to the scheduler.

For the decoding stage, video decoders relying on the FFMPEG media framework [11] and [15] are used, which provides either a hardware accelerated HEVC decoder implementation (Nvidia NVDEC for Windows [12] and MediaCodec for Android [13]) or a software implementation of the VVC decoder (OpenVVC developed by INSA/IETR [14]).

To each decoder, an array of OpenGL textures to be used as output buffer is associated. For hardware decoding, a direct GPU copy to speed up the processing is used. Once decoded, the atlases are queued toward the scheduler.

The scheduling stage waits for the atlas data and video frames needed to synthesize one frame of the content, packetize this data and waits until the frame timestamp is reached. The data is then transferred to the relevant synthesizer.

Regarding the synthesizing stage, the host is responsible for issuing the rendering command each time it wishes to render the content. It also provides the color and depth textures used as render target for the synthesizers at initialization, as well as camera and viewport information at each frame. The synthesizer then fetches the last decoded data packet and renders the content, decoupling the decoding and rendering frame rates. Using shared color and depth textures enables merging V3C content with a full 3D scene managed by the host application.

Unity Host Application

The host application manages user input, camera movement and full scene rendering. A simplified architecture of a basic Unity implementation [21] is presented in Figure 5. This implementation uses a two-camera setup: a render camera and a UI camera. The render camera draws all the 3D elements of the Unity scene into the shared colour and depth textures. The plugin then uses the camera information to render the content into the same textures. Finally, the colour buffer is rendered directly on a UI image. The second camera, set to render only the UI, produces the final render.

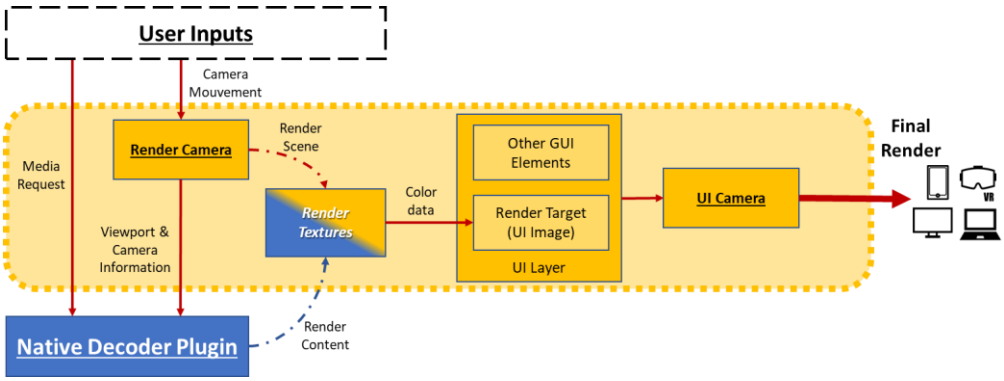

Figure 5 - High Level Unity Implementation Architecture

Architecture advantages

As shown in this section, the main implementation modules of our Immersive Video Decoder Platform are DASH streaming, dynamic video decoder(s) usage, hardware HEVC decoding with direct GPU copy, native OpenGL rendering into the synthetiser, software VVC decoding. The split architecture based on a native plugin implementation enables to reach high performance and cover multiple hosts: the plugin already supports native C++ and Unity based application and could easily be adapted to other hosts such as Unreal Engine. Finally, management of 2D video content is directly supported thanks to the video-based nature of the V3C codecs.

EVALUATION

The Immersive Video Decoder Platform evaluation was performed on Windows and Android. With Windows 11 on Dell G5 Laptop (G5) with Intel Core i7 2.60GHz and NVidia GeForce RTX 2060. With Android 13 on Samsung Tab S8 Ultra (S8) with Qualcomm SM8450 and Qualcomm Adreno 730.

The next sections describe generation of V-PCC and MIV content that feeds the Platform and results of the evaluation for each technology are given.

V-PCC Content Generation

For V-PCC, five input content items (Soccer Blue, Soccer Red, Dancer01, Acrobat01, Acrobat Duo) have been delivered by the XD productions company, a service provider company specialized in videogrammetry techniques [19]. Content screenshots are provided in Figure 6. The capture setup is composed of about sixty 4K cameras, arranged in hemispheres around the scene to be captured. The set is 15-meter in diameter for a 7-meter diameter capture area. Two types of lenses are simultaneously used, with variable focal lengths, which allows to adapt the size of the capture area, and to mix wide shots and close-ups on the same captures to improve the quality of the textures. Each content item has then been converted into point cloud frames contained in a 1024-sized bounding box with integer, positive coordinates.

Each content item is encoded using the Test Model for Category 2 based on the release R22.0 [9], at both R3 (QP geometry=24, QP Attribute=32, Occupancy precision=4) and R5 (QP geometry=16, QP Attribute=22, Occupancy precision=2) rates, as defined in CTC [7] using the profile HEVC Main10 V-PCC Basic Rec0.

Figure 6 – XD Productions contents screenshots, from top right to left: Acrobat01, Soccer Blue, Soccer Red, Dancer01, Acrobat Duo

When decoding 10-bit content on Android, a rescaling of the MediaCodec decoded video frame is done, in turn corrupting the 3D reconstruction. No issues were detected on Windows, nor using 8-bit content on Android. For easier cross platform comparison, the 8-bit content on both platforms has been chosen. The frame size of the encoded videos has been locked at 1492×1600 (1492x1920 for Acrobat01) on the encoder side to avoid dynamic updates of the frame size during the decoding process. The encoding was done with a single map in each atlas to limit the number of hardware HEVC decoders to 3, reducing the point count for multi-layer content.

V-PCC Content Name	FPS	Input average point count/frame	Input bitrate (Mbps)	Rates	Output average point count/frame	Average point count ratio	Output bitrate (Mbps)	Compression ratio
Soccer Red	25	1065014	1371,16	R3	542 214	0,51	4,74	289
				R5	505 816	0,47	15,81	87
Acrobat01	30	824 905	1062,03	R3	840 691	1,02	4,42	240
				R5	811 090	0,98	20,86	51
Soccer Blue	30	274 665	353,62	R3	280 094	1,02	3,77	94
				R5	264 958	0,96	13,76	26
Dancer01	30	311 351	400,85	R3	137 204	0,44	1,67	240
				R5	127 989	0,41	6,68	60
Acrobat Duo	30	786 132	1012,11	R3	802 133	1,02	6,02	168
				R5	765 915	0,97	26,45	38

Table 1 – V-PCC Test Model for Category 2 release 22.0 rates.

Table 1 provides a summary of key characteristics of the input content and the encoding result.

Uncompressed bitrates are computed considering a 10-bit geometry and 8-bit per-channel colour coding. Point counts ratios around 0.5 correspond to a single map being encoded for a multi-map input and to encoder choices during the atlas generation phase. Variations in output point number for the same stream at different rates is caused by the different Occupancy Precision used.

MIV Content Generation

MIV performances were evaluated using four self-captured content items:

- **Dance**: shot with 6 Azure Kinect cameras at FHD resolution, placed in a linear array spaced at an approximate 12 cm baseline, hardware synchronized and captured at a rate of 15 fps. The depth-maps, available from the Kinect were ignored.
- **Soccer**: shot with 8 uEye UI-3080 CMOS cameras at FHD resolution, spaced at an approximate 25 cm baseline, hardware synchronized and captured at a rate of 30 fps.
- **Mannequin**: Computer-Generated Images (CGI) content, recorded at 25 fps in Unity using 15 FHD virtual cameras.
- **Barn**: shot with 15 Blackmagic Micro Studio cameras at 4K resolution, synchronized using Genlock sync, captured at a rate of 30 fps and spaced at 29 cm baseline. Rigs used for Barn and Mannequin allows horizontal and vertical parallax.

Content screenshots are shown in Figure 7 for Dance and Soccer and in **Error! Reference source not found.** for Mannequin and Barn.

Figure 7 – Philips Content screenshots: Soccer, Dance

Figure 8 – Interdigital Content screenshots: Mannequin, Barn

Soccer and Dance recordings were prepared for encoding into MIV bitstreams using the MIV Extended MVD+T profile, with texture, geometry, and transparency atlases. The encoding performs the following steps: undistortion, camera extrinsic estimation, multiview depth estimation and segmentation. Internally developed tools were used for the last two steps to separately handle the (static) background and the (dynamic) foreground objects. The result of the object segmentation is a second output of this step and is used in the MIV encoding. The Dance content used an atlas frame size of 4096×2824 pixels, while the Soccer content used an atlas size of 4096×3272 pixels.

Mannequin and Barn recordings were encoded using the MIV Extended MPI profile, which uses texture and transparency atlases. Barn content is challenging due to complex scene capture and real-time depth estimation tool used, generating lots of noise in the depth maps. It has been selected as high bit rate reference to evaluate the performance of the MIV MPI processing. All atlases were encoded into HEVC bitstreams using a 32 frame inter-period and two bitrate profiles were used. Table 2 lists the contents key statistics. For the input bitrate, the input resolution times the number of cameras was used, taking 25 bits per pixel (10 for luminance, 5 for chrominance and 10 for depth). The input considered are a texture (yuv 4:2:0 10-bit LE) and a depth (yuv 4:2:0 16-bit LE) video streams. Both contents used an atlas frame size of 4096×4096 pixels.

For each content, a low rate and a high-rate encodings are generated with selected QPs for texture, geometry (if applicable) and transparency atlases.

MIV Content name	Profile	FPS	Input Resolution (HxWxN)	Encoder input bitrate (Mbps)	Rates	QPs [Text;Geom;Transp]	Encoder output bitrate (Mbps)	Compression ratio
Dance	MVD+T Extended profile	15	1920*1080*6	4449	Low	[35; 35; 35]	5,89	756
					High	[25; 25; 25]	15,98	278
Soccer	MVD+T Extended profile	30	1920*1080*8	11865	Low	[35; 35; 35]	12,55	946
					High	[25; 25; 25]	36,51	325
Mannequin	MPI Extended profile	25	1920*1080*15	35596	Low	[43; na; 27]	11	3236
					High	[39; na; 18]	15	2373
Barn	MPI Extended profile	30	1920*1080*15	42715	Low	[32; na; 44]	202	211
					High	[24; na; 44]	339	126

Table 2 – MIV Encoding Performances

Test Results

Each content is read locally and split into data chunks containing at most 32 frames and fed synchronously with the target frame rate (i.e., a chunk every 1.0666s for 30 FPS or every 1.28s for 25 FPS, …). It plays for 1 minute starting at the first decoded frame while looping if needed.

The rendering frame rate and the decoder frame rate were measured. The decoded frame rate is set as the rate at which a full set of decoded frames plus the corresponding atlas data is received, before scheduling. This explains why the decoder frame rate may exceed that of the input content. These two measures are needed as both stages are not synchronised

in our implementation. Test on Windows are realized without V-sync to render as fast as possible. On Android, the rendering is synchronized with the screen refresh rate by default (120 Hz). The results are presented in Table 3 and Table 4 for V-PCC and MIV respectively.

Stream	FPS	Rates	Average Decoder FPS		Average Renderer FPS		Rates	Average Decoder FPS		Average Renderer FPS	
			G5	S8	G5	S8		G5	S8	G5	S8
Soccer Red	25	R3	25,35	25,17	292,94	30,20	R5	25,21	25,21	280,85	30,21
Acrobat01	30	R3	30,25	30,30	239,64	30,17	R5	30,29	30,11	235,97	30,21
Soccer Blue	30	R3	30,28	30,27	262,77	30,21	R5	30,27	30,29	259,22	30,21
Dancer01	30	R3	30,25	30,30	271,14	30,21	R5	30,27	30,19	269,02	30,21
Acrobat Duo	30	R3	30,30	30,22	226,22	30,17	R5	30,35	30,23	226,48	30,19

Table 3 – V-PCC Evaluation Results

Stream	FPS	Rates	Average Decoder FPS		Average Renderer FPS		Rates	Average Decoder FPS		Average Renderer FPS	
			G5	S8	G5	S8		G5	S8	G5	S8
Dance	15	Low	15,20	15,13	205,44	30,21	High	15,20	15,17	206,18	29,99
Soccer	30	Low	30,49	**28,46**	178,60	**28,41**	High	30,34	**29,22**	179,67	**28,88**
Mannequin	25	Low	25,33	25,31	140,78	30,03	High	25,34	25,33	151,59	30,19
Barn	30	Low	30,53	**27,59**	98,74	**27,87**	High	30,55	**22,13**	109,82	30,11

Table 4 – MIV Evaluation Results

On the Windows G5 platform, all streams are decoded and rendered above the targeted FPS. On the Android S8 platform, all V-PCC and most of MIV content are also decoded and rendered to the target FPS. Some MIV content (bolded in **Error! Reference source not found. Error! Reference source not found.**) show signs of limitations with real-time decoding. This is linked to streams complexity (see Table 2) and better results might be obtained by optimizing associated decoders. The screen / rendering synchronicity explains the renderer FPS homogeneity across content on Android.

Our platform can decode V3C content in real time on both Windows and Android. This implementation proves that V3C decoding is production ready on high-end Android devices [20]. Thanks to the natural evolution of hardware, it is reasonable to assume that most devices will be able to decode V3C content in the near future.

CONCLUSION

The paper presents the first implementation of the MPEG-I V3C standards in one plug-in and one application, providing implementation insights for the industry. The proposed architecture shows the path to an integration of MPEG-I standards into the XR ecosystem with off-the shelves platforms such as the Unity framework.

The use of open standards like MPEG, demonstrates the ability to deploy such content and experience at large scale. Moreover, leveraging the compression efficiency of MPEG video codecs, the proposed solutions demonstrate viable use cases for volumetric assets and scenes. Evolution from HEVC codec to VVC codec will allow stronger compression

efficiency. Integration of V3C into a future version of the scene description standard will enable global scene compositing and more complex environment distribution. Integration of live encoding will open the path toward live teleconferencing and telepresence.

ACKNOWLEDGEMENTS

We thank Jacques Peyrache and Philippe Souchet from XD productions for content courtesy.

REFERENCES

[1] ISO/IEC 23090-5, Information technology — Coded Representation of Immersive Media — Part 5: Visual Volumetric Video-based Coding (V3C) and Video-based Point Cloud Compression (V-PCC)

[2] ISO/IEC 23090-12, Information technology — Coded Representation of Immersive Media — Part 12: MPEG immersive video

[3] ISO/IEC 23090-10, Information technology — Coded representation of immersive media — Part 10: Carriage of visual volumetric video-based coding data

[4] ISO/IEC 23090-14, Information technology — Coded representation of immersive media — Part 14: Scene description

[5] ISO/IEC JTC1/SC29 WG11, Online, April 2021, MDS20352, V-PCC codec description

[6] International Organization for Standardization/International Electrotechnical Commission (ISO/IEC) JTC1/SC29 WG11, “Use Cases for Point Cloud Compression,” Genova, Switzerland, w16331, Jun. 2016

[7] International Organization for Standardization/International Electrotechnical Commission (ISO/IEC) JTC1/SC29 WG11, “Common Test Conditions for PCC,” Alpbach, Austria, w19324, Apr. 2020

[8] SMPTE, C. Guede, P. Andrivon, J. -E. Marvie, J. Ricard, B. Redmann and J. -C. Chevet, "V-PCC Performance Evaluation of the First MPEG Point Codec," in SMPTE Motion Imaging Journal, vol. 130, no. 4, pp. 36-52, May 2021, doi: 10.5594/JMI.2021.3067962, https://ieeexplore.ieee.org/document/9424091

[9] Test Model for Category 2 (TMC2) release 22.0 for MPEG V-PCC, MPEG 142, April 2023, https://dms.mpeg.expert/doc_end_user/documents/142_Antalya/wg11/MDS22732_WG07_N00572.zip

[10] Test Model Version 14 for MPEG Immersive Video, MPEG 139, July 2022, https://dms.mpeg.expert/doc_end_user/documents/139_OnLine/wg11/MDS21853_WG04_N00242.zip

[11] FFmpeg, https://ffmpeg.org/, v5.1.1

[12] NVidia Video Codec SDK, https://developer.nvidia.com/video-codec-sdk

[13] Android MediaCodec, https://developer.android.com/reference/android/media/MediaCodec

[14] OpenVVC, real-time software decoder developed by INSA/IETR, https://github.com/OpenVVC/OpenVVC

[15] OpenVVC, FFmpeg fork with OpenVVC lib support developed by INSA/IETR, https://github.com/OpenVVC/FFmpeg

[16] International Organization for Standardization/International Electrotechnical Commission (ISO/IEC) JTC1/SC29 WG11, Call for proposals for point cloud compression v2, N16763, 2017

[17] Bart Kroon, MIV Tutorial, IEE VCIP 2021. PowerPoint Presentation (mpeg-miv.org)

[18] J. M. Boyce et al., "MPEG Immersive Video Coding Standard," in Proceedings of the IEEE, vol. 109, no. 9, pp. 1521-1536, Sept. 2021, doi: 10.1109/JPROC.2021.3062590.

[19] XD productions (https://www.xdprod.com/).

[20] MPEG Immersive Video website MPEG Immersive video (MIV) – Specification for streaming & storage of immersive content (mpeg-miv.org)

[21] Unity v2020.3.22f1 (https://unity.com/)