https://doi.org/10.7840/kics.2025.50.12.1917

ISOBMFF Encapsulation for Carriage of G-PCC Bitstream: A Tutorial and Guidelines

Seunghyeok Jeong◆, Gwanghyeon Jeong*, Dong Ho Kim**, Dongho You°

ABSTRACT

This paper presents a tutorial on encapsulating Geometry-based Point Cloud Compression (G-PCC) bitstreams within the ISO Base Media File Format (ISOBMFF). To enhance the storage and transmission efficiency of point cloud data, we analyze and compare the structural characteristics, advantages, and disadvantages of single-track, multi-track, and tile-based encapsulation methods. Additionally, we introduce a method for managing various encoded versions using the alternative track mechanism. This approach enables the optimized utilization of G-PCC data in multimedia environments. This study demonstrates the potential of ISOBMFF-based encapsulation to ensure compatibility with existing media streaming systems, while improving the efficiency of 3D data transmission and storage.

Key Words : Carriage of G-PCC, ISOBMFF Encapsulation, Overview, Perspectives

Ⅰ. Introduction

As the utilization of 3D data continues to expand, the importance of efficient storage and transmission technologies is becoming increasingly critical. Volumetric video, as a key enabler for immersive experiences in virtual, Various fields, including autonomous driving, precision mapping, as well as virtual, augmented, and extended reality (VR/AR/XR) content and medical simulations, demand more precise 3D representations[1]. For instance, in the medical field, virtual environments and 3D models of the human body provide valuable insights before actual procedures[2]. These technologies are not only beneficial for medical education but also serve as essential tools for skilled surgeons performing complex operations. Specifically, computed tomography (CT) scans and magnetic resonance imaging (MRI) data can be transformed into 3D visualizations, which are rendered in virtual environments using graphics processing units (GPUs)[3]. This enables medical professionals to conduct detailed analyses of a patient' s condition, streamline surgical planning, and enhance diagnostic accuracy and treatment strategies. Beyond healthcare, virtual reality is also actively utilized in retail and e-commerce[4]. Retailers can showcase products in virtual spaces without physically arranging them in stores, allowing customers to experience products virtually before making a purchase. In such environments, the ability to efficiently compress, store, and rapidly transmit high-resolution 3D data is essential for maximizing the effectiveness of 3D applications.

There are several methods for representing 3D data, including mesh, volume data, and point cloud[5,6]. A mesh constructs a 3D model using polygons such as triangles or quadrilaterals and is the most commonly used format in industries like computer graphics and game engines. Volume data, on the other hand, repre-

※ This research was supported by Seoul National University of Science and Technology.
◆ First Author : Hannam University, Department of Information and Communication Engineering, 20254056@m365.hnu.ac.kr, 학생회원
° Corresponding Author : Kongju National University, Department of Computer Education, dongho.you@kongju.ac.kr, 정회원
* Hanbat National University, Department of Semiconductor System Engineering
** Seoul National University of Science and Technology, Department of Smart ICT Convergence Engineering
논문번호 : 202503-059-D-RN, Received March 17, 2025; Revised June 11, 2025; Accepted July 30, 2025

sents 3D objects by defining voxels-volume pixels that encapsulate spatial information-making it particularly useful for medical imaging, scientific visualization, and data analysis. Point cloud representation is a key technology due to its ability to precisely capture objects in three-dimensional space. A point cloud consists of numerous individual points, each containing spatial coordinates (x, y, z) along with additional attributes such as color and reflectivity. Because of these characteristics, point clouds have become a widely adopted format for scanned data representation and are extensively used in applications such as environmental perception using LiDAR sensors, 3D modeling, and realtime 3D streaming.

However, as the number of points in a point cloud increases, its data size grows exponentially, making efficient storage and transmission a critical challenge[6]. Compared to conventional images and videos, point cloud data not only requires significantly larger storage but also presents irregular structures and high computational complexity. As a result, additional technologies are needed for compression and transmission. To address this issue, the Moving Picture Experts Group (MPEG) under the International Organization for Standardization/ International Electrotechnical Commission (ISO/IEC) has established standards for the efficient compression of 3D data, among which Geometry-based Point Cloud Compression (G-PCC) is a key standard[7-9]. However, compression alone is not sufficient for efficiently storing and utilizing 3D data. To seamlessly integrate 3D data with existing multimedia content, such as video, images, and audio, it is essential to leverage an appropriate file format. The ISO Base Media File Format (ISOBMFF), as defined in ISO/IEC 14496-12, provides a flexible structure for storing various multimedia data and has the potential to effectively manage 3D bitstreams such as G-PCC[10]. In this context, MPEG is in the process of standardizing the "Carriage of G-PCC", which defines how G-PCC data is packaged within ISOBMFF[11,12]. Encap-sulating G-PCC bitstreams within ISOBMFF not only ensures compatibility with existing multimedia streaming systems but also enhances the efficiency of 3D data storage and transmission. Furthermore, storing G-PCC data within ISOBMFF enables efficient navigation and access to specific 3D frames using metadata and sample tables. Additionally, the extensible file structure of ISOBMFF allows for flexible adaptation to future requirements of emerging 3D content.

This paper presents a systematic tutorial on how to encapsulate G-PCC bitstreams within the ISOBMFF, based on the specifications defined in ISO/IEC 23090-18. To support a comprehensive understanding, it also incorporates essential background concepts from ISO/IEC 23090-9 (G-PCC compression) and ISO/IEC 14496-12 (ISOBMFF structure), which are prerequisites for fully interpreting the 23090-18 specification. The paper organizes and analyzes key elements such as the storage of G-PCC data in the ISOBMFF track structure, the composition of metadata, and the structure of sample tables. In addition, it compares the strengths and limitations of each encapsulation method-including aspects not explicitly described in the standard-and presents various application scenarios. Through this, the paper aims to provide an integrated understanding of 3D data storage and transmission structures and to support the selection of suitable encapsulation methods based on specific use cases.

Ⅱ. Background

To understand how G-PCC bitstreams are encapsulated within the ISOBMFF, it is essential to first grasp the structure of the G-PCC bitstream and the foundational elements of the ISOBMFF. Specifically, ISO/IEC 23090-9 defines the compression format for Geometry-based Point Cloud Compression (G-PCC)[7], while ISO/IEC 14496-12 specifies the structural organization of the ISOBMFF[11]. These two standards form the basis for the encapsulation guidelines described in ISO/IEC 23090-18[10]. This section provides a brief overview of the G-PCC bitstream and the relevant ISOBMFF components to support a clearer understanding of the encapsulation process.

2.1 G-PCC Bitstream

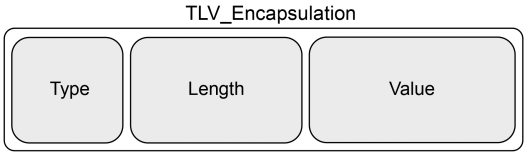
Point cloud data poses significant challenges in

storage and transmission due to its large data size. A single object can consist of an enormous number of points, and each point may include various attribute data, causing an exponential increase in data volume. Therefore, compression and optimization techniques are essential for efficiently handling point cloud data. To address this issue, the MPEG under the ISO/IEC has established compression standards, among which G-PCC is a key technology. In G-PCC, spatial coordinates and attribute information are separately compressed as Geometry Data and Attribute Data, respectively.

To efficiently store and transmit point cloud data within the ISOBMFF, it is essential to optimize the underlying data structures. Various data representation formats are used for this purpose, and when G-PCC bitstreams are employed, the Type-Length-Value (TLV) format is adopted. As illustrated in Fig. 1, the TLV format consists of three components: Type, Length, Value.

- Type: indicates the kind of information contained in the data block.
- Length: specifies the size of the Value field in bytes.
- Value: contains the actual data content.

This structure facilitates the efficient separation and processing of data with variable lengths. For example, a Type value of 0 indicates that the data block contains a Sequence Parameter Set, while a value of 2 denotes that it contains a Geometry Data Unit. A complete list of Type values and their corresponding meaning is provided in Table 1.



tlv_encapsulation() {	Descriptor
tlv_type	u(8)
tlv_num_payload_bytes	u(32)
for(*i* = 0; *i* < tlv_num_payload_bytes; *i*++)	
tlv_payload_byte[*i*]	u(8)
}	

Fig. 1. TLV unit structure.

Table 1. Value about TLV type.

Type	Description
0	Sequence Parameter Set
1	Geometry Parameter Set
2	Geometry Data Unit
3	Attribute Parameter Set
4	Attribute Data Unit
5	Tile Inventory
6	Frame boundary marker
7	Defaulted attribute data unit
8	Frame specific attribute properties data unit
9	User data data unit

The G-PCC bitstream is generally structured as illustrated in Fig. 2. It begins with the Sequence Parameter Set (SPS), which contains fundamental information such as the compression profile and the characteristics of the content. This is followed by the Geometry Parameter Set (GPS) and Attribute Parameter Set (APS),which provide the necessary configuration for decoding geometry and attribute data, respectively. The Geometry Data Unit (GDU) and the Attribute Data Unit (ADU) represent the actual point cloud data in compressed form.

These data units are consistently represented using the TLV format, where the Type value allows for easy identification of each data block. As such, TLV is highly effective for representing G-PCC data, offering flexibility, extensibility, and structural clarity.

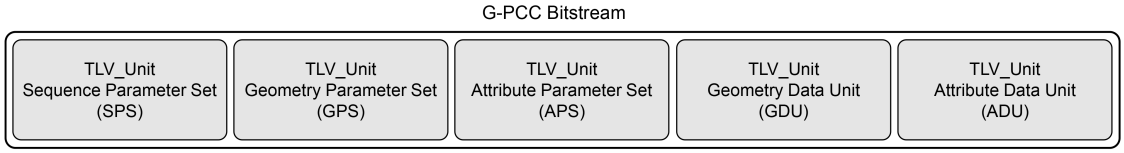


Fig. 2. G-PCC bitstream structure.

2.2 ISOBMFF Structure

The ISOBMFF is a standardized container format defined in ISO/IEC 14496-12, designed to efficiently store both time-based media data (such as audio and video) and non-time-based media data (such as static images). ISOBMFF operates on a flexible box-based structure, where all data is hierarchically organized within nested boxes(A "Box" refers to a container that encapsulates a specific type of data, functioning similarly to a folder in a file system by organizing and distinguishing different types of information). Each box serves a specific function and consists of a header and a payload. The header includes the Size field, indicating the box length, and a Four-Character Code (4CC) field, which identifies the type of box and defines its payload format. This structure allows ISOBMFF to efficiently manage various media types within a single file.

An ISOBMFF file is composed of multiple key boxes, as illustrated in Fig. 3, and typically includes the following core components:

- File Type Box (ftyp): Specifies the standard and compatibility of the file by defining supported brands.
- Movie Box (moov): Contains metadata that defines the overall structure of time-based media. Within the moov box, individual media streams are organized into Track Boxes (trak), where each track represents a different media type (e.g., video, audio, text, or other data).
- Media Data Box (mdat): Stores the actual media content, including raw audio and video data.
- Metadata Box (meta): Manages non-time-based media, such as static images. Unlike time-based media, where metadata is stored in the trak box inside the moov box and actual data is stored in the mdat box, non-time-based media uses the meta box for metadata storage, while actual content is placed in the mdat box or the Item Data Box (idat).

In ISOBMFF, time-based media data is stored in units called samples, which represent individual data elements such as a single frame of video or a specific duration of audio. A key concept in managing samples is the sample entry, which defines codec information and data format within a track, specifying how the data should be decoded and processed. Each track can contain multiple sample entries, which are stored in

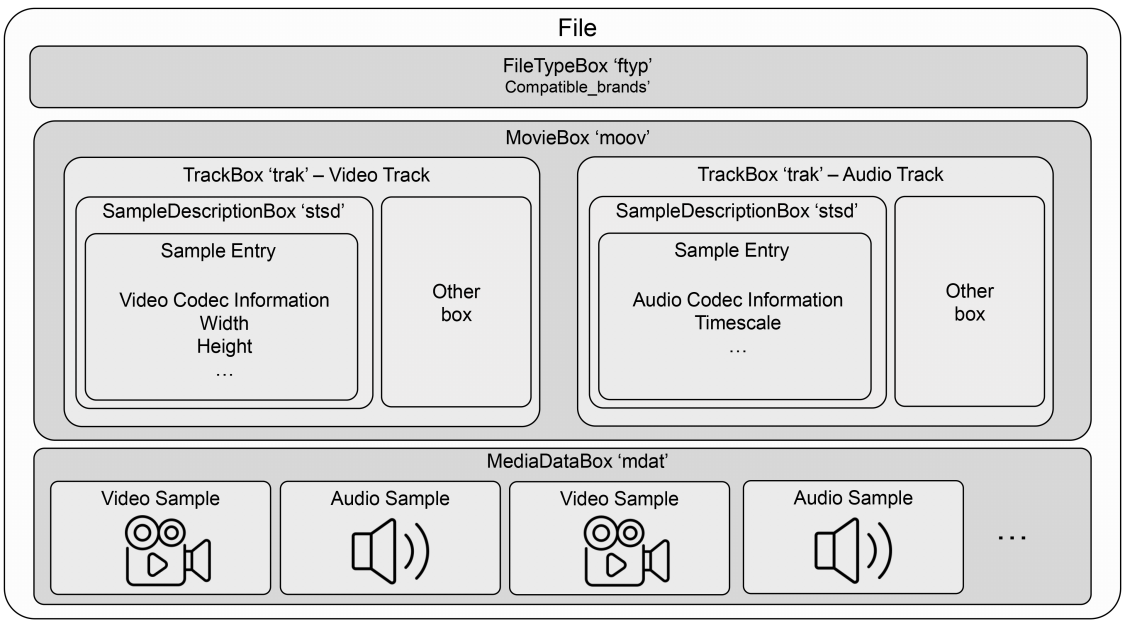


Fig. 3. Example of basic ISOBMFF file structure.

Table 2. Key elements of the ISOBMFF-based format used in this paper.

Name	Type	Description
ftyp	box	File type box, defines the file format and compatibility
moov	box	Movie box, includes the overall media structure and metadata
mdat	box	Media data box, stores the actual media data
trak	box	Track box, stores individual media streams
tkhd	box	Track header box, includes track ID, duration, etc.
hdlr	box	Handler box, specifies the handler type for the track
stsd	box	Sample description box, contains sample entry information
gpcC	box	G-PCC configuration box, defines decoder level settings such as profile and level
ginf	box	G-PCC component information box, describes geometry or attribute components
ctts	box	Composition offset box, specifies the composition time by providing offsets
stts	box	Time to sample box, specifies decoding time intervals using sample count and delta pairs
meta	box	Meta box, includes metadata for non-timed content
idat	box	Item data box, stores static data
iprp	box	Item properties box, manages item properties
pitm	box	Primary item box, specifies the primary item
gpst	brand	Brand identifier for single-track G-PCC encapsulation
gpmt	brand	Brand identifier for multi-track G-PCC encapsulation
gpci	brand	Brand identifier for non-timed G-PCC track encapsulation
gpe1	sample entry	Used for single-track encapsulation; SPS/GPS/APS appear only in decoder config
gpeg	sample entry	Used for single-track encapsulation; SPS/GPS/APS may appear in config and sample
gpc1	sample entry	Used for multi-track encapsulation; SPS/GPS/APS appear only in decoder config
gpcg	sample entry	Used for multi-track encapsulation; SPS/GPS/APS may appear in config and sample
gpeb	sample entry	Used when geometry and attribute data are encapsulated together in the same tile track
gpcb	sample entry	Used when geometry and attribute data are stored separately in different tile tracks
gpt1	sample entry	Used in G-PCC tile track
gpca	track reference	Used when the referenced track is an attribute track
gpbt	track reference	Used when the referenced track is a G-PCC Tile track

the Sample Description Box (stsd). Meanwhile, non-time-based media data is stored in items, which serve as the fundamental storage units for static content. The key elements of the ISOBMFF-based format used in this paper-including boxes, brand identifiers, sample entries, and track reference types are summarized in Table 2.

Ⅲ. ISOBMFF Encapsulation for Carriage of G-PCC Bitstream

All encapsulation methods described in this section(single-track, multi-track, and tile-based track) utilize common elements of the ISOBMFF structure to carry G-PCC bitstreams. Across these methods, each track that carries point cloud data uses the following conventions:

- The compatible_brands field in the File Type Box (ftyp) must indicate the encapsulation type used in this file. This value helps the decoder determine the encapsulation structure.
- The handler_type in the Handler Box (hdlr) must be set to volv to indicate volumetric content. This

informs the media system that the track contains volumetric point cloud data.

- The sample entry must be one of the extended volumetric types and include a compressor_name field that specifies the compression format. Each entry also contains G-PCC Configuration Box (gpcC). This box provides the configuration and parameter set information necessary for decoder initialization.
- The gpcC box holds a decoder configuration record with the following fields:
  - configurationVersion: Specifies the version of the decoder configuration record.
  - profile_compliant: Defines the profile that the bitstream complies with.
  - level_idc: Represents the performance level of the decoder.

The gpcC box also includes a setupUnit array, which stores immutable parameter sets in TLV format. These parameter sets consist of SPS, GPS, and APS. The exact contents of the setupUnit array depend on the encapsulation method, as described in each subsection.

3.1 Single-Track Encapsulation

The single-track encapsulation method involves storing the entire encoded point cloud sequence within a single track. This approach allows for the straightforward storage of a G-PCC bitstream in ISOBMFF without requiring additional preprocessing or demultiplexing. To indicate single-track usage, the compatible_brands field in the ftyp box must be set to gpst. The sample entry must be of type gpe1 or gpeg. These formats differ in where the parameter sets are included.

- In gpe1, all parameter sets must be included in the setupUnit array and reused across all samples. This format is efficient for content with fixed configurations, as it simplifies decoder initialization and reduces per-sample overhead (As shown in Fig. 4).
- In gpeg, parameter sets may be included in the setupUnit array or embedded directly in each sample. This format provides flexibility for dynamic content, as it allows mid-sequence configuration updates without modifying the global setup (As shown in Fig. 5).

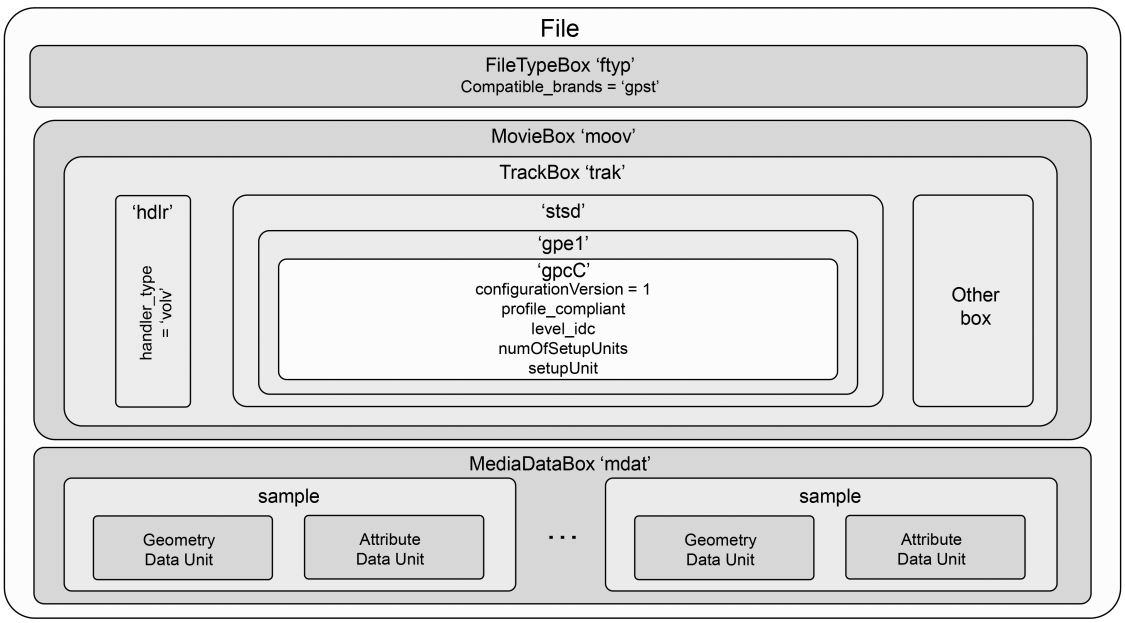


Fig. 4. Sample entry structure of gpe1.

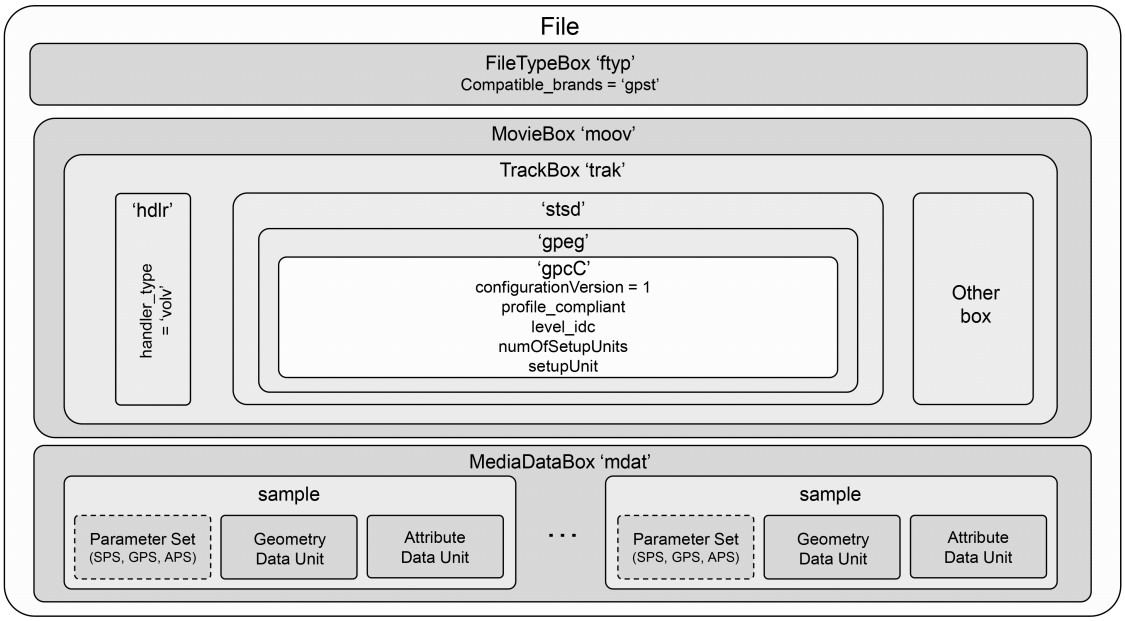


Fig. 5. Sample entry structure of gpeg.

Each sample in the track represents a single point cloud frame and must include at least one Geometry Data Unit (GDU). Attribute Data Units (ADU) may be present if required by the content structure.

3.2 Multi-Track Encapsulation

The multi-track encapsulation method separates geometry and attribute components of a G-PCC bitstream into distinct tracks. This structure enables independent access, parallel decoding, and selective data management. To indicate multi-track usage, the compatible_brands field in the ftyp box must be set to gpmt. Each component (geometry or attribute) is stored in its own track. The sample entry must be of type gpc1 or gpcg. These formats differ in where parameter sets are included.

- gpc1 requires parameter sets to be included in the setupUnit array: SPS and GPS in geometry tracks, and APS in attribute tracks. (As shown in Fig. 6).
- gpcg allows parameter sets to either be included in the setupUnit array or embedded within each samples. (As shown in Fig. 7).

gpc1 and gpcg adopt a similar configuration handling strategy to gpe1 and gpeg, balancing between reuse and per-sample flexibility.

Each sample must only contain data from a single component. Therefore, GDU and ADU cannot be mixed in the same sample. Furthermore, if the bitstream contains multiple attribute components, each must be assigned to a separate attribute track to maintain separation and avoid interleaving.

In addition to the gpcC box, each sample entry in multi-track encapsulation must include a G-PCC Component Information Box (ginf). This box contains a comp_type field that specifies whether the track

holds geometry or attribute data. A value of 2 indicates a geometry component, while a value of 4 designates an attribute component. In the latter case, the following additional fields are required:

- attr_index: Identifies the attribute component and specifies its sequence order as defined in the SPS.
- attr_label: Identifies the attribute component type.
- attr_name: Stores a human-readable name for the attribute component.

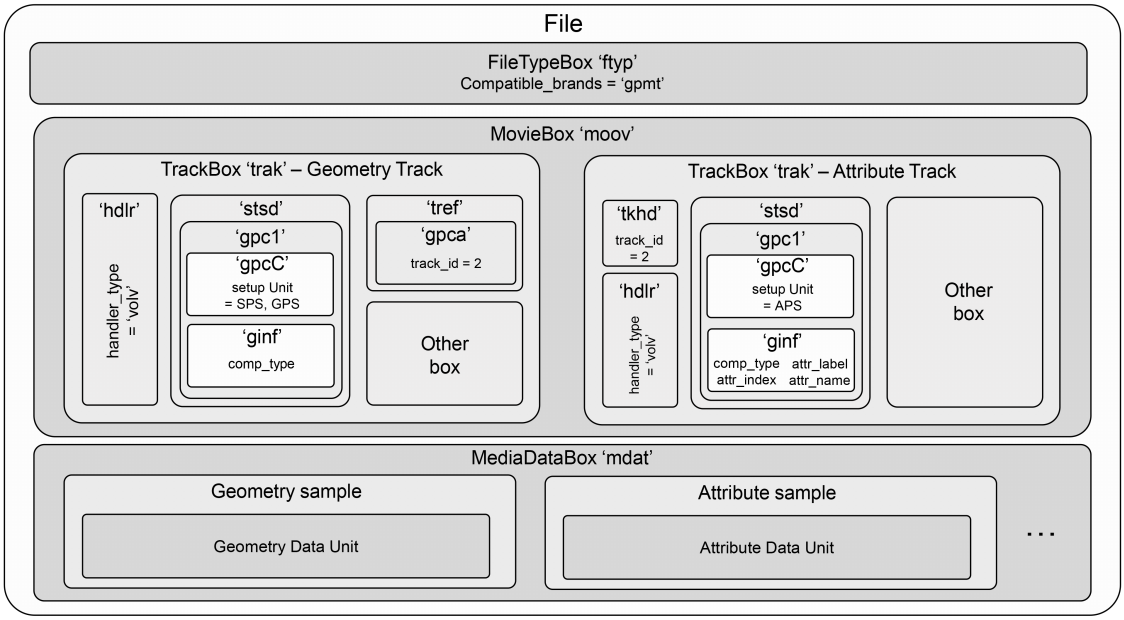


Fig. 6. Sample entry structure of gpc1.

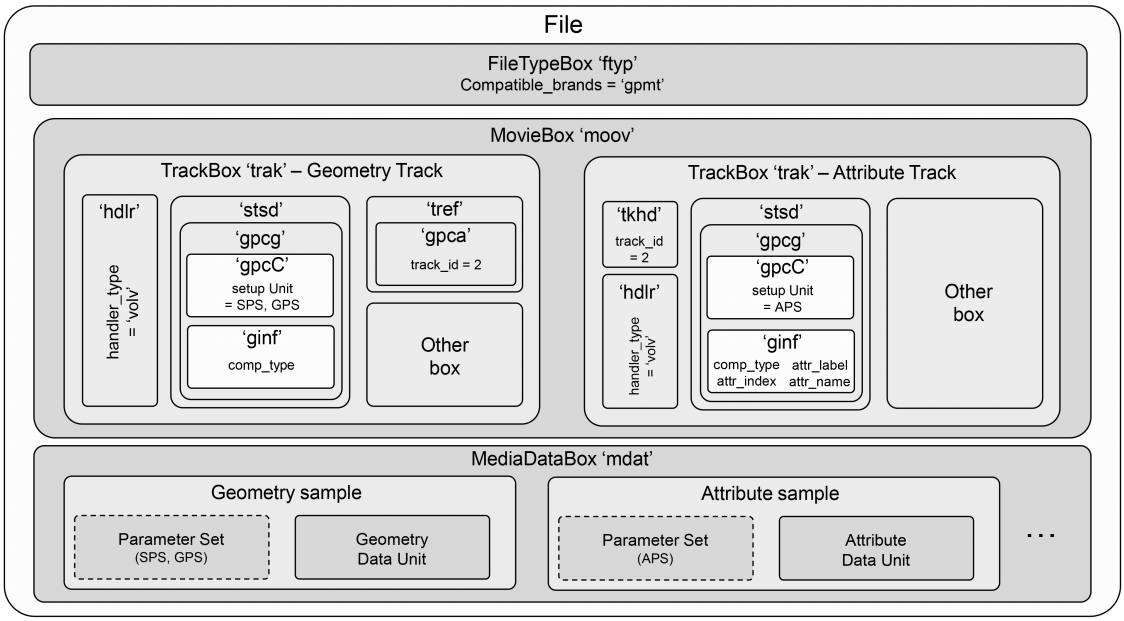


Fig. 7. Sample entry structure of gpcg.

All component tracks must share the same profile_compliant and level_idc values. The geometry track must also reference each associated attribute track using a Track Reference Box (tref) with reference type gpca, containing the referenced track ID.

To ensure frame synchronization, samples representing the same point cloud frame across different tracks must have identical presentation times. Parameter sets must be decoded no later than the data units they configure. In cases where SPS, GPS, or

APS are distributed across multiple tracks, samples containing SPS must be decoded before those containing GPS or APS.

By separating geometry and attribute data, the multi track encapsulation approach offers flexible and efficient access. Clients can independently retrieve and decode only the components they require. However, if a bitstream includes only geometry data and no attributes, the single-track encapsulation method must be used instead.

3.3 Tile-Based Encapsulation

The tile-based encapsulation method partitions a G-PCC bitstream into tiles and encapsulates them across multiple tracks in ISOBMFF. This approach enables independent access and decoding of G-PCC data for specific tile regions. This method uses one tile base track and multiple tile tracks.

- The tile base track contains global parameter sets (SPS, GPS, APS) and a tile inventory.
- It references each tile track using a Track Reference Box(tref) with reference type gpbt. Each tile track must use a sample entry of type gpt1.

The tile base track uses either gpeb or gpcb as its sample entry type:

- gpeb stores all components (geometry and attributes) per tile within a single track, similar to single-track encapsulation (As shown in Fig. 8).
- gpcb separates geometry and attribute components into different tile tracks, similar to multi-track encapsulation (As shown in Fig. 9). Geometry tile tracks may reference attribute tile tracks via gpca. In such cases, the track_in_movie flag of the attribute tile track must be set to 0.

In tile-based encapsulation, each sample in the tile base track contains SPS, GPS, APS, and tile inventory data. Samples must include synchronized parameter set data referenced by the tile base track, and all tile tracks must maintain the same presentation time. This synchronization is managed using the Composition Offset Table (ctts) and Time-to-Sample Table (stts) of the tile base track, ensuring precise decoding of specific tiles or components.

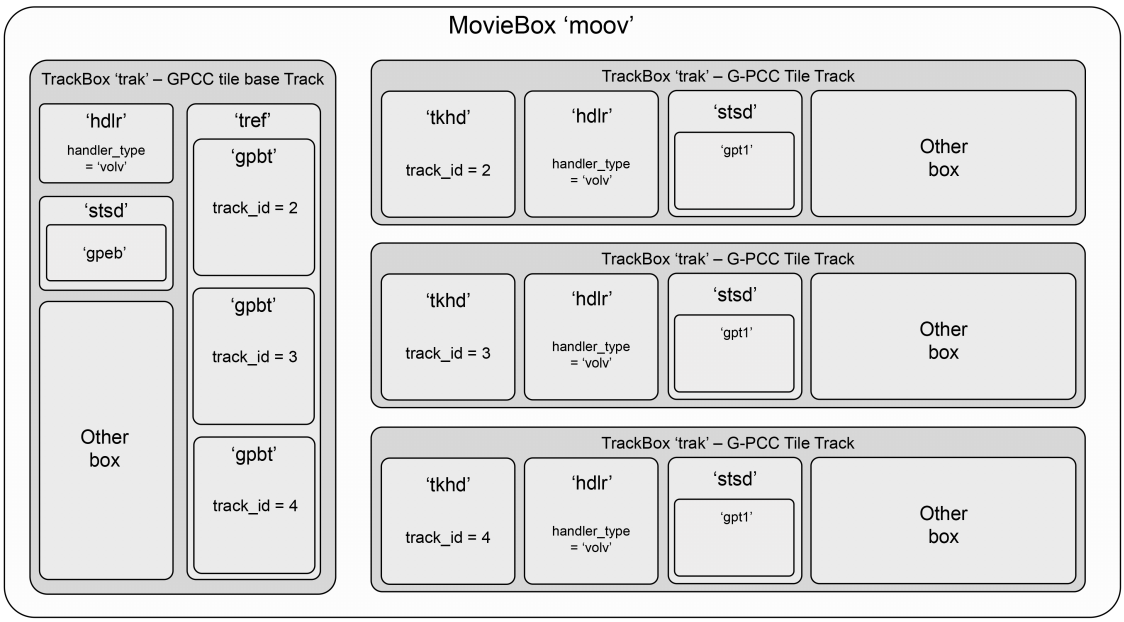


Fig. 8. moov box structure when using gpeb sample entry.

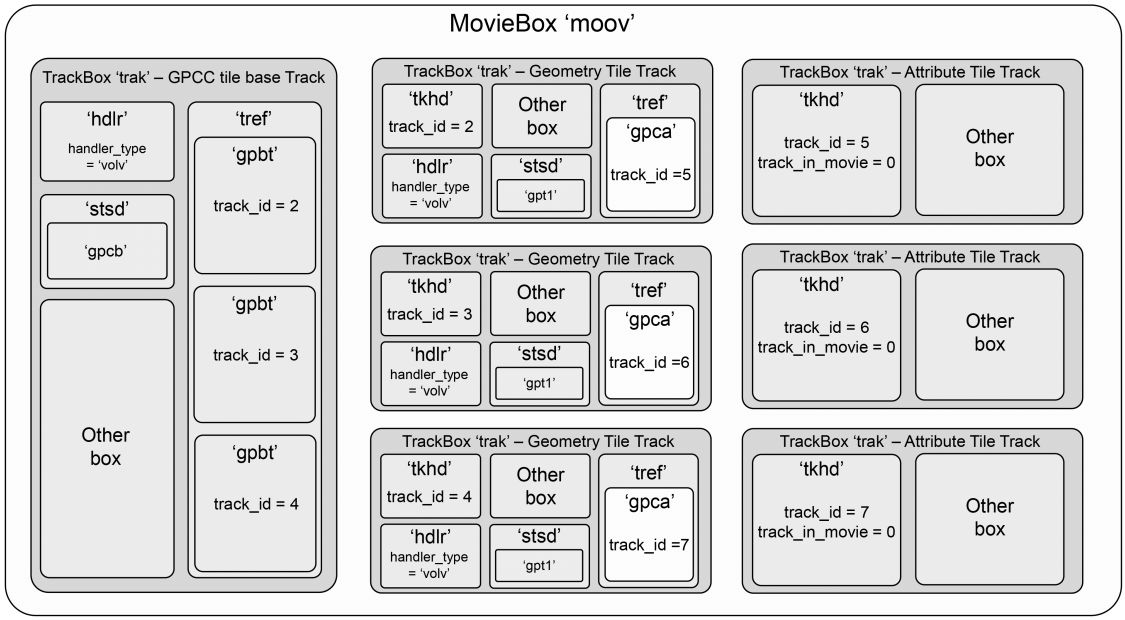


Fig. 9. moov box structure when using gpcb sample entry.

3.4 Track Alternatives

A point cloud sequence can be stored in multiple encoding formats to provide optimal quality and performance across various environments. Considering factors such as network conditions, storage capacity, and rendering performance, it is possible to encode a G-PCC bitstream into multiple versions, each representing the same point cloud sequence in a different manner.

In ISOBMFF, an alternative track mechanism is provided to efficiently manage these alternative encoded versions. The alternative tracks are grouped using the alternate_group field in the Track Header Box (tkhd), allowing multiple G-PCC tracks representing the same point cloud sequence to coexist while enabling the selection of an appropriate track based on the playback environment.

- Without Tile Tracks
  - Each version of the same point cloud (e.g., lossy vs lossless) is stored as a separate geometry track.
  - All such tracks must share the same alternate_group value in the Track Header Box.
  - Only one track from the group should be selected for playback (As shown in Fig. 10).
- With Tile-Based Tracks
  - Each version consists of a tile base track and corresponding tile tracks.
  - All tile base tracks for alternative version must share the same alternate_group value in the Track Header Box.
  - Fig. 11 illustrates how tile base and tile tracks are grouped for selection during playback.

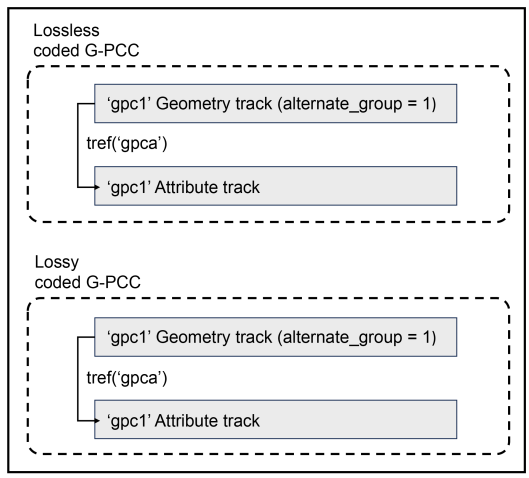


Fig. 10. Track alternatives indication in multi track of G-PCC Bitstream.

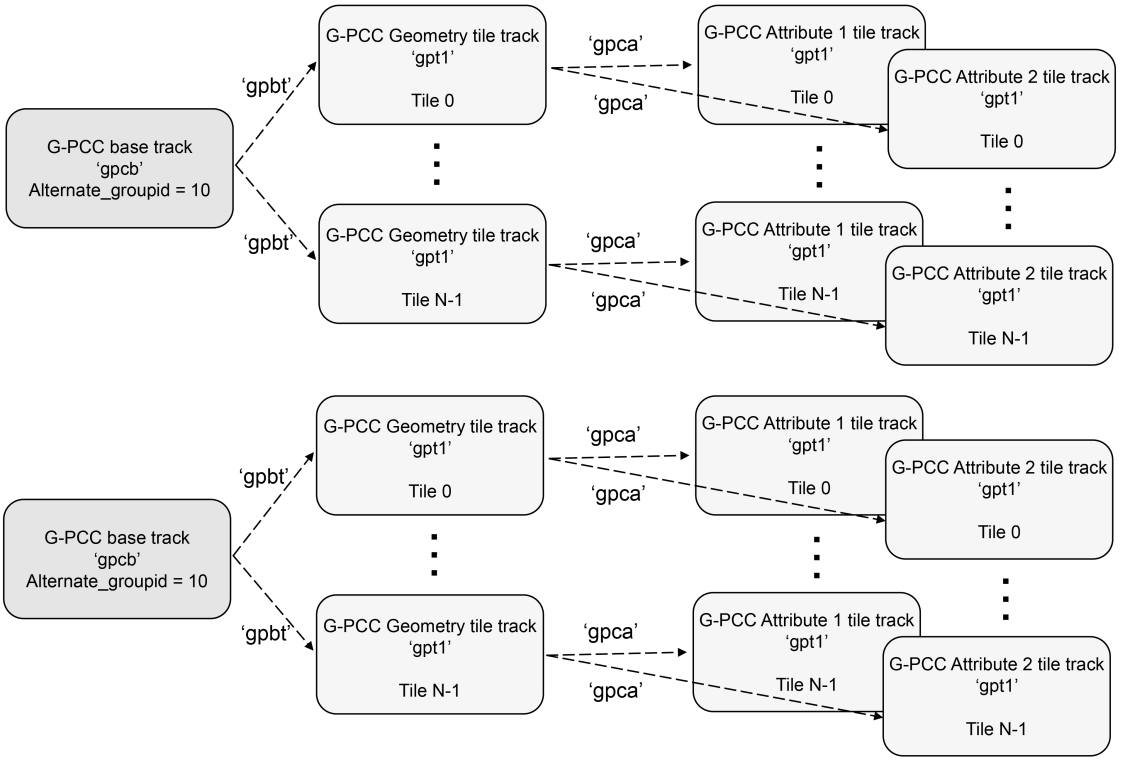


Fig. 11. Track alternatives indication in tile base track of G-PCC Bitstream.

3.5 Non-Timed Data Encapsulation

The non-timed data encapsulation method represents a G-PCC bitstream for a single point cloud frame as an item within the ISOBMFF. Unlike the track encapsulation method discussed earlier, this approach stores data as items rather than tracks, without any time information. Consequently, time synchronization is not required. The bitstream is stored in the Metadata Box (meta) and described using item properties in the Item Property Box (iprp), while the actual data is placed in the Item Data Box (idat) or Media Data Box (mdat). To indicate this method, the compatible_brands field in the ftyp box must be set to gpci, and the handler_type in the hdlr box inside meta box must be set to volv. Depending on whether the G-PCC bitstream is tiled or not, different item types are used for non-timed encapsulation. In the case of non-tiled content, a frame can be encapsulated as a single item or split into multiple items. For tiled content, a combination of a base item and tile items is employed. The following subsections describe the item types and their structure accordingly.

3.5.1 Item Types for Non-Tiled Bitstreams

Two item types are used depending on how the G-PCC bitstream is organized:

- gpe1: Used when the entire frame is stored as a single item. This item includes all data units for one point cloud frame.
- gpc1: Used when the bitstream is split across multiple items. (As shown in Fig. 12).
- When using the gpc1 item type:
  - Each gpc1 item must contain a GDU.
  - ADUs may be stored in separate gpc1 items, which must be marked as hidden.
  - GDU item reference ADU items using gpca item references.
  - If a pitm box is present, its item_ID must reference the gpc1 item with the GDU.

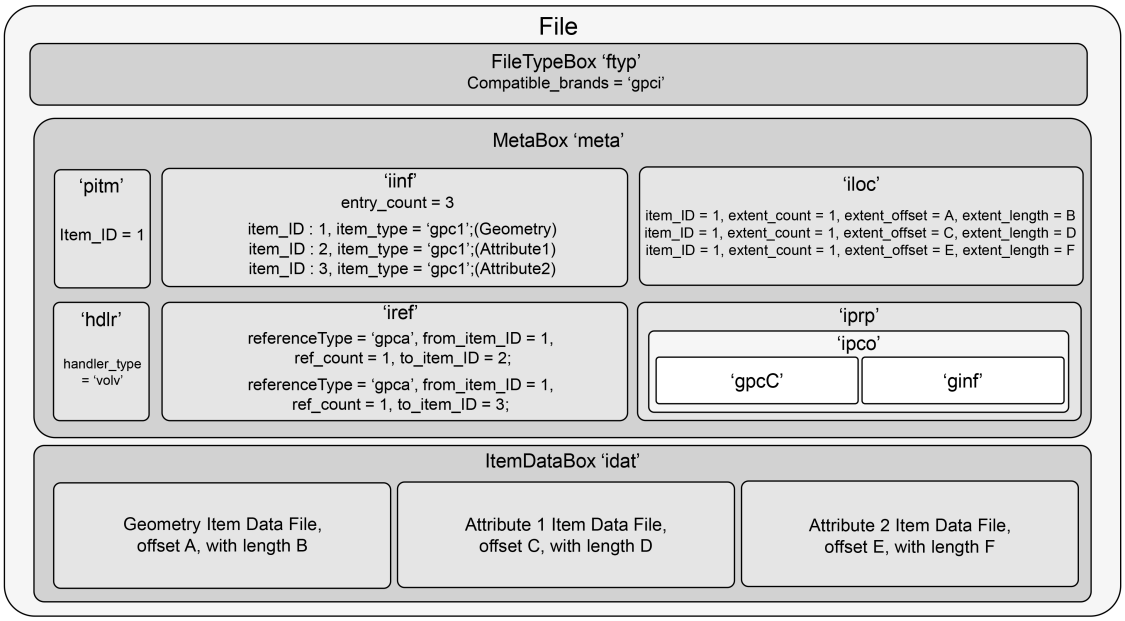


Fig. 12. Non-Timed data structure stored in multiple items.

3.5.2 Tile-based item encapsulation

If the bitstream includes tiles:

- gpeb items must store only parameter sets or tile inventory (no GDU or ADU).
- At least one gpt1 item must be present, containing tile data.
- A gpbt item reference connects the gpeb item to each gpt1 item (As Shown in Fig. 13).
- Each gpt1 item must include GDU and ADU, but not parameter sets or tile inventory.
- If a pitm box is present, its item_ID must refer-

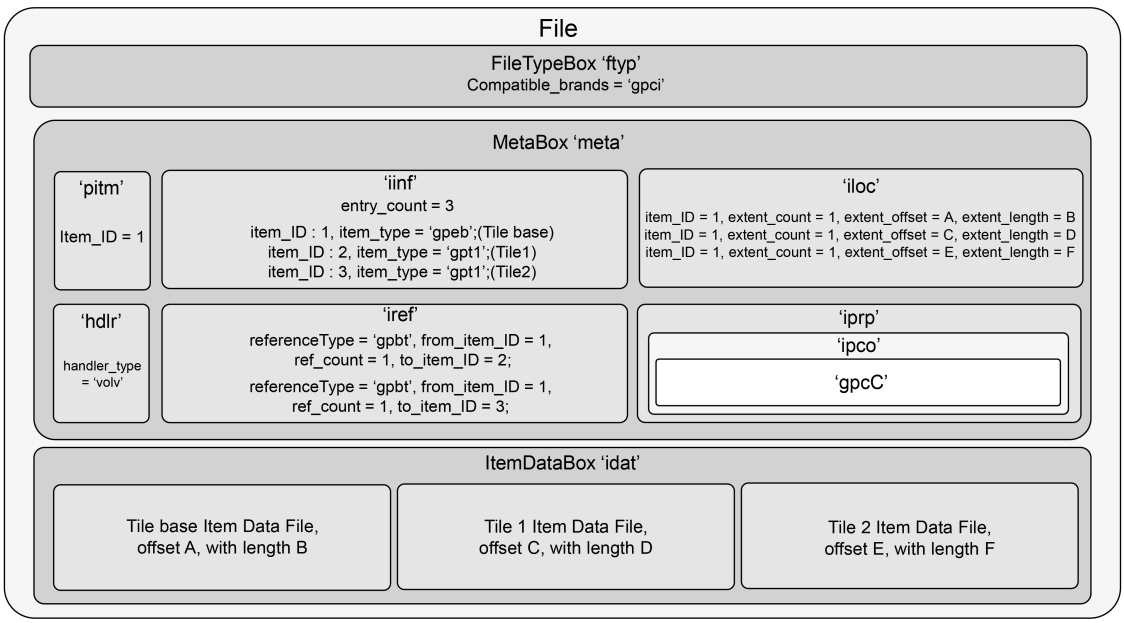


Fig. 13. Non-Timed data structure stored in tile based items.

ence the gpeb item.

Ⅳ. Perspectives and Guidelines

4.1 Technical Comparison of Encapsulation Method

The G-PCC bitstream can be encapsulated in the ISOBMFF structure using three main methods: single-track, multi-track, and tile-based track encapsulation. Each method has distinct technical characteristics in terms of structural organization, processing approach, and scalability, which may affect the selection depending on the application scenario. A summary of these characteristics is provided in Table 3.

4.1.1 Single-Track Encapsulation

The single-track encapsulation method stores all components of the G-PCC bitstream within a single track. This approach features a simple structure and straightforward implementation, eliminating the need for inter-track synchronization and thereby allowing for a clear processing flow. It is particularly advantageous for early-stage system design or resource constrained environments. The following summarizes the performance characteristics of the single-track encapsulation method:

- Memory usage: Since all components are bundled into a single sample, the entire data must be loaded into memory during decoding. This means that even if only geometry information is needed, attribute data is also loaded, leading to increased memory usage as the per-frame data complexity grows.
- Decoding complexity: While the structural simplicity facilitates easy implementation, the limited feasibility of parallel processing between geometry and attribute data restricts performance scalability.
- Synchronization overhead: Since all data is managed within a single track, synchronization can be achieved using existing ISOBMFF timing mechanisms such as stts and ctts. As inter-track alignment is unnecessary, the implementation burden related to synchronization is reduced, and the risk of timing mismatches during playback is minimized.
- Extensibility: While the simple structure allows for easy initial implementation, the integration of all components into a single track imposes significant limitations when replacing specific data or adding new attributes.

4.1.2 Multi-Track Encapsulation

The multi-track encapsulation method stores the components of the G-PCC bitstream in separate tracks based on their roles, such as geometry and attribute data. This structure enables flexible data access and facilitates parallel processing by leveraging the independence of each track, making it advantageous for complex content or high-performance environments. The following outlines the performance characteristics of the multi-track encapsulation method:

Table 3. Comparison of G-PCC encapsulation methods based on technical criteria.

Encapsulation Method	Memory Usage	Decoding Complexity	Synchronization Overhead	Extensibility
Single-track	All components loaded at once. High memory usage even for partial data.	Simple structure. Limited parallelism.	No inter-track sync. Low overhead.	Difficult to modify due to monolithic structure.
Multi-track	Selective loading improves efficiency.	Allow parallelism possible, but adds complexity.	Requires sync between tracks. Sync overhead and mismatch risk.	Easy to modify via separate tracks.
Tile-based track	Enables ROI decoding. Memory-efficient.	Allow parallel decoding; tile coordination is complex.	Tile-level sync needed. Overhead increases with tile count.	Easy per tile updates and customization.

- Memory usage: The multi-track structure enables selective access to individual tracks, allowing for more flexible memory management. For instance, during rendering, only the geometry track may be loaded, while certain attribute tracks can be skipped. This reduces unnecessary data loading and contributes to more efficient memory usage.
- Decoding complexity: Since the components are separated, each track can be independently decoded, enabling parallel processing. This is particularly advantageous in high-resolution point cloud rendering or high speed playback scenarios using multi-core CPU environments. However, as separate decoding is required for each track, the initial implementation complexity tends to be relatively high.
- Synchronization overhead: As the geometry and attribute tracks must be synchronized along a common timeline, additional mechanisms are required to ensure proper alignment. This increases the overall complexity of implementation and may elevate the risk of timing discrepancies.
- Extensibility: Since each component is stored in a separate track, it is relatively easy to add new attributes as separate tracks or to replace only specific tracks when necessary.

4.1.3 Tile-Based Track Encapsulation

The tile-based encapsulation method partitions G-PCC data spatially and stores each tile as an individual sample. This structure enables spatial parallel processing and selective decoding, making it well suited for high-resolution content and region-of-interest (ROI) based streaming[13,14]. The following outlines the performance characteristics of the tile-based encapsulation method:

- Memory usage: By spatially partitioning a frame and storing each tile as an independent sample, only the required tiles can be selectively loaded during decoding. This reduces overall memory consumption. In particular, memory efficiency is significantly improved when processing only regions of interest in high-resolution point clouds.
- Decoding complexity: Since each tile is stored as an independent sample, the structure allows for parallel decoding, which can improve processing speed if properly implemented. However, additional processing may be required to handle tile boundaries and maintain consistency. Furthermore, as the number of tiles increases, the complexity of managing tile-level decoding also grows.
- Synchronization overhead: Since the tiles collectively form a single frame aligned on the same timeline, temporal synchronization is required. This necessitates additional logic, and as the number of tiles increases, the complexity of the synchronization mechanism may also grow.
- Extensibility: The tile-based track approach allows for updating or replacing data in specific spatial regions and enables assigning different quality levels or transmission priorities to individual tiles. This provides a high degree of scalability and flexibility for various use cases.

4.1.4 Conceptual Estimation of Memory Usage

This section provides a conceptual estimation of memory usage for each encapsulation method based on actual G-PCC bitstream sizes. The comparison focuses on the conceptual characteristics of encapsulation structures, as decoding complexity and synchronization overhead can vary significantly depending on implementation details and hardware architecture.

Memory usage is estimated as the total size of data units that must be loaded into memory for decoding, depending on the encapsulation method. Specifically, we sum the sizes of the GDU and relevant ADU required for decoding.

- Single-track encapsulation: A single sample contains both geometry and attribute components. The entire sample must be loaded into memory.
- Multi-track encapsulation: Separate samples from different tracks (e.g., GDU and selected ADU) are loaded together as needed.
- Tile-based encapsulation: A set of samples corresponding to the required tiles is loaded, allowing

In this estimation, a representative point cloud containing 729,133 points was used. The G-PCC bitstream was encapsulated into a GDU and two ADUs, both representing color data but encoded using different methods: ADU1 was generated using PCM encoding, while ADU2 was generated using hierarchical neighborhood prediction as lifting transform. This allows a conceptual comparison of memory usage under different attribute coding configurations.

The size of each encapsulated data unit is summarized in Table 4. This table presents the sizes of the GDU, ADU1, and ADU2 in both all-in-one form and tile-based partitioning. Based on these data sizes, memory usage was estimated for three representative decoding scenarios: (1) loading all units (GDU + ADU1 + ADU2), (2) loading GDU and ADU1 only, and (3) loading GDU only. These scenarios reflect typical use cases depending on the application and encapsulation method.

Memory usage values were calculated by summing the sizes of the GDU and relevant ADU samples required for each scenario. In the single-track case, all components (GDU + ADU1 + ADU2) are assumed to be loaded together as they are encapsulated within a single sample. In the multi-track case, only the necessary tracks are loaded according to the decoding requirement. For tile-based encapsulation, gpeb follows the single-track strategy but allows selective tile loading of specific tiles in this estimation, tiles 1 through 4 are assumed to be loaded to represent a typical ROI scenario. In contrast, gpcb follows the multi-track strategy and enables selective loading of both attributes and tiles under the same ROI assumption.

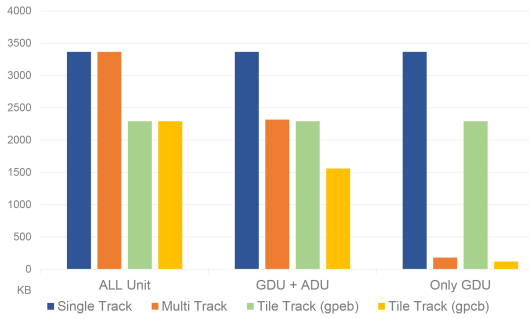


Fig. 14. Estimated memory consumption of G-PCC encapsulation methods

Fig. 14 illustrates the estimated memory consumption for each encapsulation method across the three scenarios. The results clearly demonstrate the multi-track and tile-based encapsulations provide memory-saving advantages, particularly when selective decoding is possible. These benefits are expected to become even more pronounced not only as richer attributes such as reflectance or material properties are included but also as the number of points increases in large-scale point cloud datasets.

4.2 Guidelines for Selecting Encapsulation Methods

To efficiently utilize G-PCC data, the appropriate encapsulation method should be selected based on the usage environment. Each encapsulation method has different advantages and disadvantages depending on the characteristics and requirements of the data. Therefore, it is important to determine the optimal method based on the application domain and system

Table 4. Size of encapsulated data units for GDU, ADU1, and ADU2.

Name	Size	Name	Size	Name	Size
GDU_ALL	180KB	ADU1_ALL	2,137KB	ADU2_ALL	1,048KB
GDU_Tile_1	24KB	ADU1_Tile_1	287KB	ADU2_Tile_1	132KB
GDU_Tile_2	45KB	ADU1_Tile_2	559KB	ADU2_Tile_2	275KB
GDU_Tile_3	9KB	ADU1_Tile_3	84KB	ADU2_Tile_3	47KB
GDU_Tile_4	40KB	ADU1_Tile_4	511KB	ADU2_Tile_4	279KB
GDU_Tile_5	36KB	ADU1_Tile_5	393KB	ADU2_Tile_5	165KB
GDU_Tile_6	12KB	ADU1_Tile_6	140KB	ADU2_Tile_6	67KB
GDU_Tile_7	18KB	ADU1_Tile_7	165KB	ADU2_Tile_7	81KB

Table 5. Guidelines for Selecting Encapsulation Methods

Encapsulation Method	Suitable Environment	Key Advantages	Example Use Cases
Single-track	Resource-constrained systems, simple viewers	Simplified structure, no inter-track sync, fast decoding	• Mobile web streaming with basic attributes • Lightweight AR guides or web-based viewers
Multi-track	High-performance systems, modular processing needs	Attribute separation, supports selective and parallel decoding	• Cloud XR collaboration platforms • Virtual shopping systems with separate geometry and attribute tracks
Tile-based	Large-scale or ROI based systems with selective decoding requirements	Enables regional decoding and scalable delivery	• VR exhibitions decoding only the user's current field of view • City-scale point cloud systems with per-user selective streaming

architecture. In particular, ISOBMFF is a format optimized for media-centric scenarios, such as streaming-based delivery, track-level decoding structures, and interactive media where content dynamically adapts based on user interaction[15]. Thus, careful consideration of the encapsulation method is required for such applications. A key consideration for selecting encapsulation methods in different application scenarios are summarized in Table 5.

4.2.1 Single-Track and Multi-Track Encapsulation

Selecting between single-track and multi-track encapsulation methods depends on the usage environment. The decision should be based on factors such as device performance, data processing methods, and end-user requirements.

Single-track encapsulation is suitable for resourceconstrained environments. It simplifies processing by storing all data in a single sample, eliminating the need for inter-track synchronization and enabling fast and stable decoding.

- In mobile web streaming, geometry and basic attributes such as color are encapsulated in one track to reduce bandwidth usage.
- Smartphone-based AR guide applications or lightweight web-based viewers benefit from simplified structures that enable quick content access.

Multi-track encapsulation is appropriate for high-performance environments. It supports parallel decoding by separating geometry, color, and reflectivity into distinct tracks, improving scalability and processing efficiency.

- Cloud-based XR collaboration systems allow users to selectively decode only necessary attributes from separate tracks, optimizing performance.
- In virtual shopping platforms, product geometry, texture, and reflectivity can be delivered through separate tracks, allowing flexible decoding based on device capabilities and network conditions.

Fig. 15 illustrates how these methods correspond to different device performance levels and processing architectures.

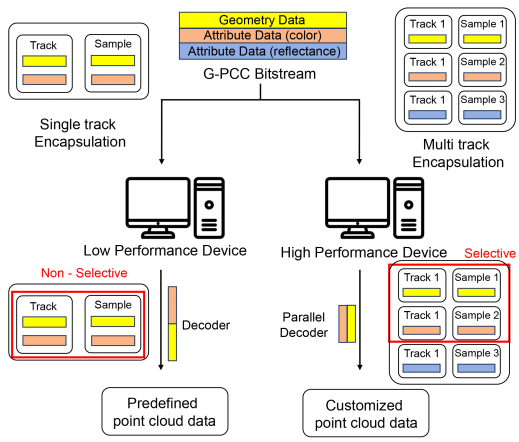


Fig. 15. Example of selecting between single-track and multi-track encapsulation based on device capabilities.

4.2.2 Tile-Based Track Encapsulation

Tile-based track encapsulation partitions G-PCC

data into tiles, with each tile stored in a separate track. This structure allows for flexible handling of large-scale point cloud data, as individual tiles can be selectively decoded based on the user's request or viewpoint, enabling efficient data transmission and processing without loading the entire dataset.

This method is particularly effective in scenarios where network bandwidth is limited or where only certain regions need to be processed according to the user's field of view.

- In a virtual reality-based 3D exhibition platform, users may only need to decode exhibition areas that fall within their current location or viewing direction. This selective decoding enables quick response and high rendering performance without unnecessary data loading.
- In smart city 3D map services, tiles corresponding to the user or vehicle's current location can be streamed in real time, allowing efficient handling of massive city-scale point cloud data.

In environments where multiple users or various viewpoints exist simultaneously, tile-based tracks offer significant advantages for selective transmission and rendering.

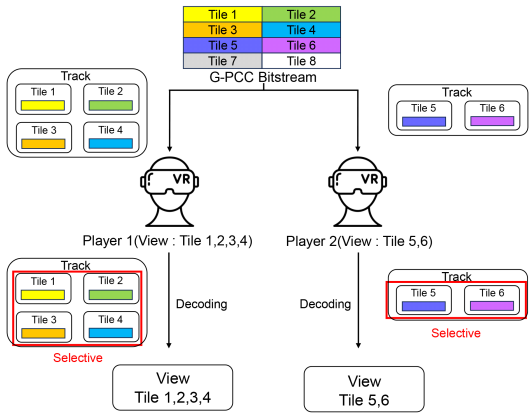


Fig. 16. Example of tile-based track encapsulation based on field of view.

As illustrated in Fig. 16, assume Player 1 views Tiles 1-4, while Player 2 views Tiles 5 and 6. In this case, only the relevant tile data is transmitted to each player. By omitting the transmission of unnecessary data, bandwidth consumption is reduced, and performance degradation due to redundant decoding is avoided. An example of applying point cloud data to real images is shown in Fig. 17.

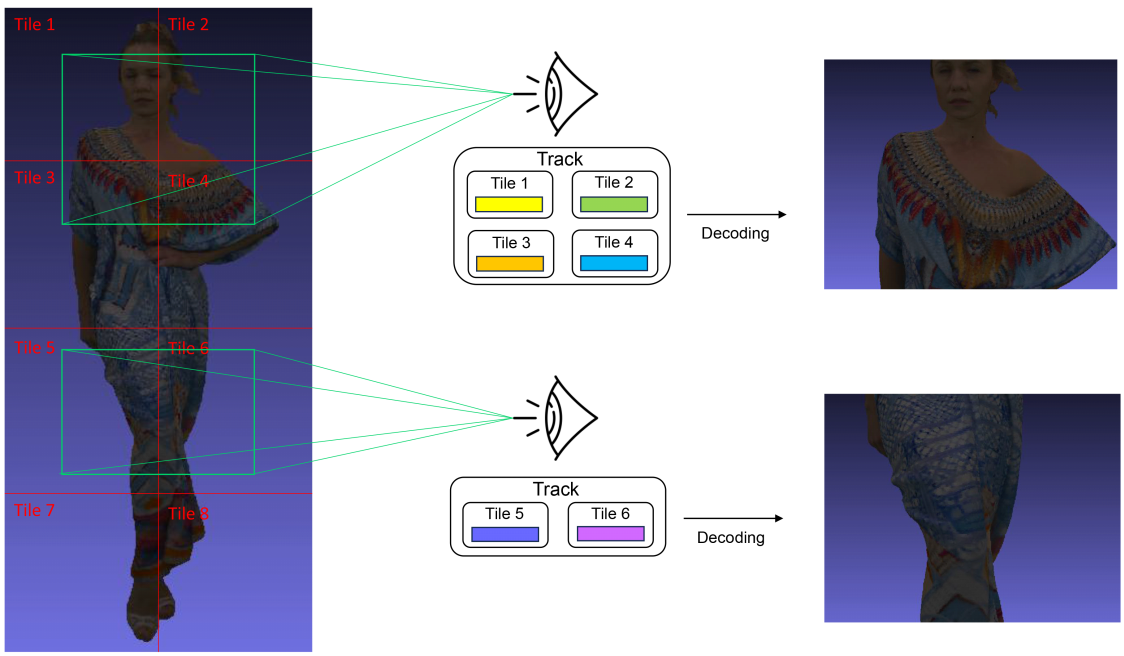


Fig. 17. Example of tile-based track encapsulation for point cloud data.

For tile-based tracks, two sample entry types are available:

- gpeb sample entry type: The dataset is divided into tiles, but geometry and attribute data are stored together within a single track, similar to single-track encapsulation.
- gpcb sample entry type: The dataset is divided into tiles, and geometry and attribute data are separately stored in different tracks, similar to multi-track encapsulation.

The choice between these two tile-based approaches follows the same selection criteria as described in Section 4.2.1, for single-track and multi-track encapsulation.

Ⅴ. Conclusion

This paper explored various methods for encapsulating G-PCC bitstreams within the ISOBMFF. We analyzed and compared the single-track, multi-track, and tile-based encapsulation methods, discussing their structural characteristics, advantages, and limitations. Additionally, we introduced an efficient management approach using the alternative track mechanism, enabling the effective handling of multiple encoded versions.

Furthermore, we provided guidelines for selecting the appropriate encapsulation method based on different usage scenarios. In performance-constrained environments, single-track encapsulation is more suitable, as it allows data to be processed sequentially without track referencing, enabling fast and simple decoding. On the other hand, in high-performance environments, multi-track encapsulation is preferable, as it facilitates parallel decoding and enables selective data access. For large-scale data processing or bandwidth-limited network environments, tile-based encapsulation is the optimal choice, as it allows selective decoding of specific tiles, thereby reducing memory usage and computational overhead.

While MPEG has developed reference implementations of the G-PCC decoder and renderer as part of the ongoing standardization process, there is currently no player capable of directly decoding and playing G-PCC bitstreams encapsulated within the ISOBMFF structure. This highlights the need for future research on developing such a player to support MP4-based G-PCC content. Moreover, a comparative evaluation of single-track, multi-track, and tile-based encapsulation methods in practical playback environments is essential to assess their respective trade-offs in performance, memory usage, and scalability.

The ISOBMFF-based G-PCC encapsulation methods proposed in this study provide a standardized approach for 3D data storage and streaming, facilitating applications in VR, AR, and digital twins. We hope that this research serves as a foundation for future advancements in G-PCC encapsulation standards and practical implementations.

References

[1] Z. Liu, Q. Li, X. Chen, et al., "Point cloud video streaming: Challenges and solutions," *Netw. Mag. Global Internet working.*, vol. 35, no. 5, pp. 202-209, Sep. 2021. (https://doi.org/10.1109/MNET.101.2000364)

[2] H. S. Chun, "Application of virtual reality in the medical field," *Electr. and Telecommunications Trends*, vol. 34, no. 2, pp. 19-28, 2019. (https://doi.org/10.22648/ETRI.2019.J.340203)

[3] Y.-J. Jung, "Development of CT/MRI based GUI software for 3D printer application," *J. Radiological Sci. and Technol.*, vol. 41, no. 5, pp. 451-456, 2018. (https://doi.org/10.17946/JRST.2018.41.5.451)

[4] H. Chun, M. Han, and J. Jang, "Application trends in virtual reality," *Electr. and Telecommunications Trends*, vol. 32, no. 1, pp. 93-101, 2017. (https://doi.org/10.22648/ETRI.2017.J.320110)

[5] A.-Y. Kim, E.-B. An, and K.-D. Seo, "Design and implementation of a point cloud-based volumetric video player," *J. KICS*, vol. 47, no. 10, pp. 1660-1668, 2022. (https://doi.org/10.7840/kics.2022.47.10.1660)

[6] P. Enenche, D. H. Kim, and D. You, "On the

road to the metaverse: Point cloud video streaming: Perspectives and enablers," *ICT Express*, vol. 11, no. 1, pp. 93-104, 2025. (https://doi.org/10.1016/j.icte.2024.11.001)

[7] International Organization for Standardization (ISO) and International Electrotechnical Commission (IEC), *MPEG-I Part 9: Geometry-based Point Cloud Compression*, Available online: https://www.iso.org/standard/78990.html, 2023.

[8] D. Graziosi, O. Nakagami, S. Kuma, A. Zaghetto, T. Suzuki, and A. Tabatabai, "An overview of ongoing point cloud compression standardization activities: Video-based (V-PCC) and geometry-based (G-PCC)," *APSIPA Trans. Signal and Inf. Process.*, vol. 9, e13, 2020. (https://doi.org/10.1017/ATSIP.2020.12)

[9] S. Schwarz, M. Preda, V. Baroncini, et al., "Emerging mpeg standards for point cloud compression," *IEEE J. Emerging and Sel. Topics in Circuits and Syst.*, vol. 9, no. 1, pp. 133-148, 2019. (https://doi.org/10.1109/JETCAS.2018.2885981)

[10] International Organization for Standardization (ISO) and International Electrotechnical Commission (IEC), *MPEG-I Part 18: Carriage of Geometry-based Point Cloud Compression Data*, Available online: https://www.iso.org/standard/80901.html, 2024.

[11] International Organization for Standardization (ISO) and International Electrotechnical Commission (IEC), *MPEG-4 Part 12: ISO Base Media File Format*, Available online: www.iso.org/standard/83102.html, 2022.

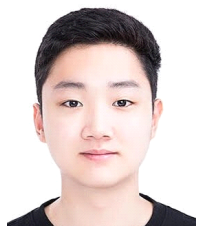
[12] M. M. Hannuksela and S. Deshpande, "VVC for immersive video streaming," in *Proc. 2nd Mile-High Video Conf.*, pp. 52-58, 2023, ISBN:9798400701603. (https://doi.org/10.1145/3588444.3591004)

[13] S. Subramanyam, I. Viola, J. Jansen, E. Alexiou, A. Hanjalic, and P. Cesar, "Evaluating the impact of tiled user-adaptive real-time point cloud streaming on VR remote communication," in *Proc. 30th ACM Int. Conf. Multimedia*, pp. 3094-3103, Lisboa, Portugal, 2022. (https://doi.org/10.1145/3503161.3548220)

[14] K. K. Sreedhar, A. Aminlou, M. M. Hannuksela, and M. Gabbouj, "Viewport-adaptive encoding and streaming of 360-degree video for virtual reality applications," in *Proc. 2016 IEEE ISM*, pp. 583- 586, 2016. (https://doi.org/10.1109/ISM.2016.0126)

[15] W. Hamidouche, L. Bariah, and M. Debbah, "Immersive media and massive twinning: Advancing toward the metaverse," *IEEE Commun. Mag.*, vol. 62, no. 7, pp. 20-32, 2024. (https://doi.org/10.1109/MCOM.2300399)

Seunghyeok Jeong



Feb. 2025 : B.Eng. Hannam University

Mar. 2025~Current : Master's Student, Hannam University

<Research Interest> Volumetric Video Streaming.

[ORCID:0009-0005-5152-5856]

Gwanghyeon Jeong

Feb. 2012 : B.Eng. Korea Advanced Institute of Science and Technology (KAIST)
Feb. 2014 : M.Eng. Korea Advanced Institute of Science and Technology (KAIST)
Feb. 2018 : Ph.D. Korea Advanced Institute of Science and Technology (KAIST)
Feb. 2018~May 2018 : Post-doctoral Researcher, KAIST Institute for Information Technology Convergence
May. 2018~Jan. 2021 : Senior Researcher, Agency for Defense Development
Mar. 2021~Feb. 2024 : Assistant Professor, Hannam University
Mar. 2024~Current : Assistant Professor, Hanbat National University
<Research Interest> Future Transceiver, MMIC, Integrated Circuits
[ORCID:0000-0002-4458-9991]

Dong Ho Kim

Feb. 1997 : B.Eng. Yonsei University
Feb. 1999 : M.Eng. Korea Advanced Institute of Science and Technology
Feb. 2004 : Ph.D. Korea Advanced Institute of Science and Technology
Mar. 2004~Feb. 2007 : Senior Researcher, Samsung Advanced Institue of Technology
Mar. 2007~Current : Professor, Seoul National University of Science and Technology
<Research Interest> Wireless Communication System, Immersive Multimedia Communication.
[ORCID:0000-0001-9136-8932]

Dongho You

Feb. 2012 : B.Eng. Seoul National University of Science and Technology
Feb. 2014 : M.Eng. Seoul National University of Science and Technology
Aug. 2018 : Ph.D. Seoul National University of Science and Technology
Sep. 2018~Feb. 2021: Senior Researcher, Technische Universität Dresden
Mar. 2021~Aug. 2025 : Associate Professor, Hannam University
Sep. 2025~Current : Associate Professor, Kongju National University
<Research Interest> Immersive Communication, Volumetric Video Streaming
[ORCID:0000-0003-3724-3244]