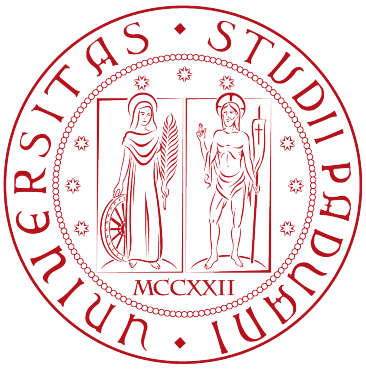

Università degli Studi di Padova

Dipartimento di Ingegneria dell'Informazione

Master Degree in ICT for Internet and Multimedia

Title

Immersive media compression using MPEG Immersive Video (MIV) standard

Supervisor:
Prof.ssa Federica Battisti

Co-Supervisors:
Prof. Marco Cagnazzo
Prof. Giancarlo Calvagno

Candidate:
Mahmoud Z. A. Wahba
2049582

Academic Year 2022-2023

Abstract

The MPEG Immersive Video (MIV) standard, plays an essential role in the coding and representation of immersive media content. This standard, designated as Part 12 within the MPEG-I suite, is invented to support virtual reality and extended reality with Six Degrees of Freedom (6DoF). MIV is designed to work on two formats, Multiview plus depth (MVD) which is the scene captured from multi-cameras arranged in different locations, and Multi Planar (MPI/MSI) which is a representation of a 3D scene that involves multiple layers. MIV standard design to exploit traditional 2D video coding like Advanced Video Codec (H.264), High-Efficiency Video Coding (H.265), and Versatile Video Coding (H.266) which had been developed over two decades. To make it applicable, the input views need to cross multi-steps to be encoded using these 2D techniques. Muti views needed to store efficiently with a new format called Atlases. To make it applicable, the input views need to label as basic views and additional views. Basic views which can be packed as it is in the atlases, while for additional views, a process to delete the redundancy between input views needs to be performed using pruning processes to generate the patches, then these patches will be inserted in Atlases after basic views. And finally, the atlases can be encoded using 2D traditional video coding. MIV provides 3 main profiles to meet diverse user requirements: MIV Main Profile, MIV Extended Profile and Geometry Absent Profile. This research provides an in-depth exploration of the entire MIV workflow, offering both theoretical insights and practical perspectives, with a notable emphasis on the MIV Main Profile. Concluding the research with a crucial phase of quality assessment, which is an essential aspect of any encoding standards evaluation to ensure the alignment with anticipated outcomes.

Acknowledgements

I would like to take this opportunity to express my deep appreciation and gratitude to the individuals who have played an indispensable role in the successful completion of my thesis.

First and foremost, I am profoundly thankful to my parents, Naima and Zakari, for their unwavering love, support, and encouragement throughout my academic journey. Your belief in me has been a constant source of motivation, and I am truly blessed to have you as my pillars of strength.

I am indebted to my esteemed supervisors, Professor Federica Battisti, Professor Marco Cagnazzo, and Professor Giancarlo Calvagno, for their invaluable guidance, insightful feedback, and continuous support. Your expertise and mentorship have been instrumental in shaping the direction of my research and enhancing the quality of my work.

I extend my heartfelt gratitude to my brothers, sisters, and friends for their unwavering encouragement and understanding during the challenging phases of this endeavor. Your presence and words of encouragement have provided me with the determination to persevere and excel.

To all those who have contributed in various ways, big and small, to this thesis, I am truly thankful. Your support, advice, and encouragement have been essential to the successful completion of this academic milestone.

Lastly, I would like to express my gratitude to the academic institution for providing me with the resources and opportunities necessary for the fulfillment of my academic aspirations.

Thank you all for being an integral part of my journey and for helping me achieve this significant accomplishment.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Background and motivation

Modern technologies such as virtual reality (VR), Extended reality (XR), and 360-degree videos, have seen significant growth in recent years, these technologies are pushing the boundaries of immersivity, allowing users to have a new experience, and to see virtual worlds as they were physically present with a high degree of freedom. To represent immersive content, many approaches can be used. Noteworthy among these there are techniques such as point cloud representation, light field, and MultiView plus depth (MVD).

However, as we delve into the realm of immersive content, the challenges of data size expansion become evident. The seeking of effective and efficient solutions to leverage constrained bandwidth while preserving content quality has captivated the efforts of researchers and corporations alike. Many compression standards have emerged, with many primarily focusing on the compression of two-dimensional videos. Recently a new family of standards had been founded called MPEG-I which stands for Moving Picture Experts Group – Immersive which is a suite of standards developed by the MPEG group that specifically address immersive media technologies.

MPEG-I had many standards, for example, MPEG-I Part 5, which is Visual Volumetric Video-based Coding (V3C) and Video-based Point Cloud Compression (V-PCC) which focuses on compressing point cloud data, which is a set of points arranged in space that are produced when objects or volumetric shapes have been scanned in 3 dimensions using LiDAR or similar type of technologies. MPEG-I part 12 is the MPEG Immersive Video (MIV) standard which was developed to support the compression of immersive video content, in which a real or virtual 3D scene is captured by multiple real or virtual cameras.

The representation and compression of immersive and volumetric data offer a diverse landscape for exploration. A lot of work was done on how to compress volumetric data to utilize the bit-stream effectively without losing the quality on the decoder side to achieve a high Quality of Experience (QoE) and user satisfaction. Increasing the quality of experience is crucial in immersive multimedia streaming. Users expect high-resolution videos, smooth playback, and minimal latency to feel fully immersed in virtual worlds and to feel the immersivity with Six Degrees of Freedom (6DoF) without any limitations. By utilizing advanced streaming techniques, such as adaptive streaming algorithms and efficient video codecs, we can optimize the use of available bandwidth and deliver a superior QoE.

Both MIV and V-PCC enable efficient compression and transmission of immersive content, maximizing the quality of the viewing experience while utilizing bitstream efficiently. MIV and V-PCC are exploiting traditional 2D video coding in their processes after applying some transformation of input data. However, in this thesis, the MIV standard will be studied as a study case. We will investigate deeply in MIV standard, with MVD format which multi cameras capture the same scene from different locations.

1.2 Objectives

Objectives of the Thesis:

- Objective 1: Understand the concept of immersive media and explore various potential methods for representing volumetric data and investigate the ways which can be used to compress these volumetric data.
- Objective 2: Acquire a comprehensive understanding of the MIV standard, with a specific focus on the Test Model introduced by the standard. Detailed analysis of the encoding and decoding processes will be undertaken.
- Objective 3: Implement practical compression, decompression, and rendering procedures using TMIV software proposed by MIV standard.
- Objective 4: Perform a quality assessment on the outcomes produced by the TMIV software, evaluating the extent to which the MIV standard effectively achieves data compression and decompression while maintaining quality.

1.3 Outline of the thesis

The thesis will follow the outlined structure as follows:

- Chapter 2: This chapter delves into the fundamentals of video compression for Immersive media. It begins by explaining the concept of immersive media and subsequently explores various methods to represent immersive content, such as light field, point cloud, and MVD. Furthermore, the chapter defines volumetric video coding and explains the diverse techniques utilized for compressing volumetric data.

- Chapter 3: Here, the focus shifts to the MIV standard, with particular emphasis on the Test Model (TMIV) proposed within the standard. The chapter provides a comprehensive breakdown of the entire pipeline encompassing encoding and decoding processes. Additionally, it delves into the specifics of the MIV bitstream and concludes with an explanation of the Common Test Conditions proposed by the MIV standard.

- Chapter 4: This chapter is dedicated to the implementation aspect. It begins by detailing the sequences chosen for implementation, followed by a comprehensive walkthrough of the processes of encoding, decoding, and rendering using the TMIV software, which is proposed by the MIV standard.

- Chapter 5: Quality assessment takes center stage in this chapter, with a comprehensive definition and explanation of the process. The objective quality evaluation is conducted on the results obtained in Chapter 4. The MIV standard's IV-PSNR is employed for evaluation purposes, and the VMAF quality metric is also applied to measure the perceptual quality.

- Chapter 6: The final chapter serves as a conclusion to the thesis, summarizing the undertaken work. It also offers recommendations for future work in this field.

Chapter 2

Fundamentals of Video Compression for Immersive Media

2.1 Introduction

In recent times, new technologies have emerged in the market, such as virtual reality (VR), mixed reality (MR), and augmented reality (AR). These cutting-edge techniques have been developed to transport users into immersive virtual worlds with a high degree of realism and freedom, eliminating boundaries and limitations. The power of immersive media lies in its ability to provide an unparalleled level of immersion, surpassing traditional methods of content consumption. Various approaches have been utilized to implement immersive content, including point cloud representation, MVD, and several other methods.

In this chapter, we will begin by defining immersive media in Section 2.2, exploring the concept and significance of creating immersive experiences for users. In Section 2.3, we will delve into different types of representations used to create immersive content, such as point cloud and MVD, highlighting their unique characteristics and advantages. Finally, in Section 2.4, we will discuss various techniques employed for coding volumetric data.

2.2 What is Immersive Media?

In the world of multimedia, the camera system plays a crucial role. Its origins can be traced back to 1816 when Joseph Nicéphore Niépce invented the first recognized camera. The main purpose of this invention was to capture visual scenes from the real world, preserving images for future use. Initially, these images were stored on photographic films, but with technolog-

ical advancements, they transitioned into digital formats. Today, images and videos can be easily captured, stored, compressed, and transmitted in real time, enabling the existence of telepresence. This means that people can feel fully present and even perform actions remotely from a distant location. This progress has led to the emergence of a significant term called ”immersion”, which refers to the feeling of being present in a virtual world [1]. Immersion is vital for creating a sense of virtual presence, making any technology that aims to give users this experience known as immersive technology.

Immersive technology provides users with a heightened sense of immersion and a greater degree of freedom (DoF) compared to traditional 2D screens (Figure 1.1) [2]. When using a 2D screen, users are confined to the viewpoint provided by the content creator or photographer, resulting in zero degrees of freedom. Conversely, with immersive technology, users can experience more than three degrees of freedom (3DoF). This means they can change their point of view through rotational movements, such as altering yaw, roll, and pitch angles. Additionally, some immersive technologies offer even more freedom, known as 3DoF plus (3DoF+), where limited spatial movement is allowed in addition to the rotational 3DoF. The ultimate level of freedom is achieved with 6 degrees of freedom (6DoF), which enables users to move in three-dimensional space through both translation and rotation. This includes changes in yaw, roll, and pitch, as well as movements up/down, forward/back, and left/right. Immersive technology enhances the user's experience by providing a dynamic and interactive environment, allowing them to explore and interact with digital content in a way that was not possible with traditional 2D screens [3].

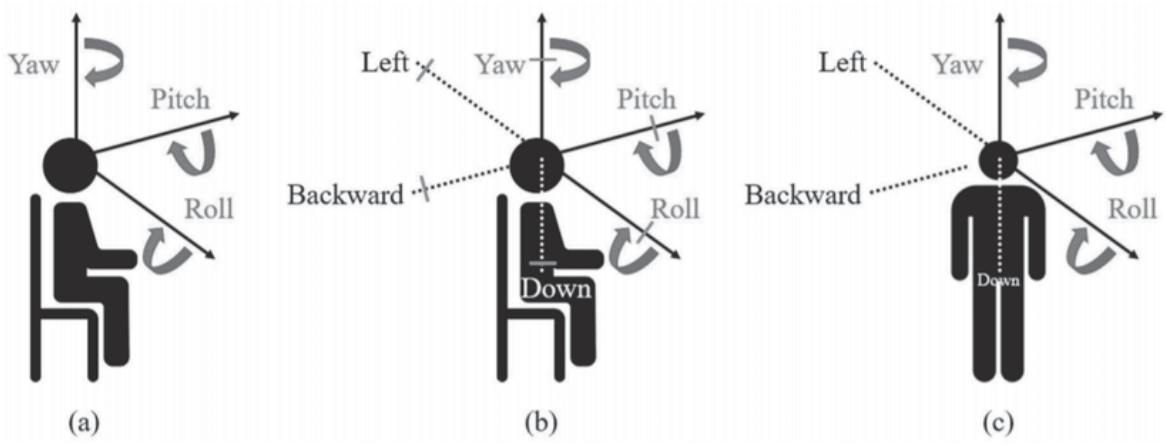

Figure 2.1: Viewing angle and degrees of freedom in VR: (a) 3-DoF, (b) 3-DoF+, (c) 6-DoF

2.3 Representation for Immersive Media

Immersive media can be created through various methods to achieve the desired level of immersion, and multiple representations have been developed for this purpose. One such representation is the light field techniques which use for example a light field camera that captures the light rays emanating from a scene to create immersive content. Another approach involves representing 3D objects using a solid representation in 3D space to create immersive content, where the basic building blocks are called voxels. Alternatively, 3D objects can be represented through polygon meshes, a common approach in computer graphics where faces typically consist of triangles or other shapes to construct 3D objects. The point cloud technique is another representation method, where 3D objects are represented as a collection of points in 3D space. This approach is often used for efficient storage and rendering of large-scale 3D environments. A more recent technique involves using multiple cameras to capture images of the same object from different viewpoints. These views, containing textures and depth information, are then used to reconstruct the 3D content, resulting in a compelling immersive experience. In this section, I will explore these representations for immersive video in greater detail.

2.3.1 Light Field

The concept behind light field technology is to capture a vast amount of optical information and then use post-processing techniques to achieve 3D visualization. Unlike traditional cameras that only capture the intensity of light, light field cameras also record the direction of incoming light, providing greater flexibility to reconstruct images from different viewpoints. Light field photography, introduced by Ng and Levoy in 2005 [4], aims to capture the light field and process the acquired image information. Various methods have been proposed to capture the light field, including using camera arrays with slight shifts between each camera to obtain a 4D light field which was proposed by Stanford University [5]. Another approach involves using light field cameras with microlens, as suggested by Ng [4] and Adelson [6]. These cameras use a microlens array between the main lens and the sensor, enabling post-processing to reconstruct different views using the captured light data from these microlenses. The key advantage of light field technology lies in its ability to create more realistic and interactive virtual environments. By manipulating the captured light rays, viewers can change their perspective within the scene, focus on different objects, and experience a higher level of presence in the virtual space. Examples of Light Field acquisition are shown in Fig. 2.2 [7].

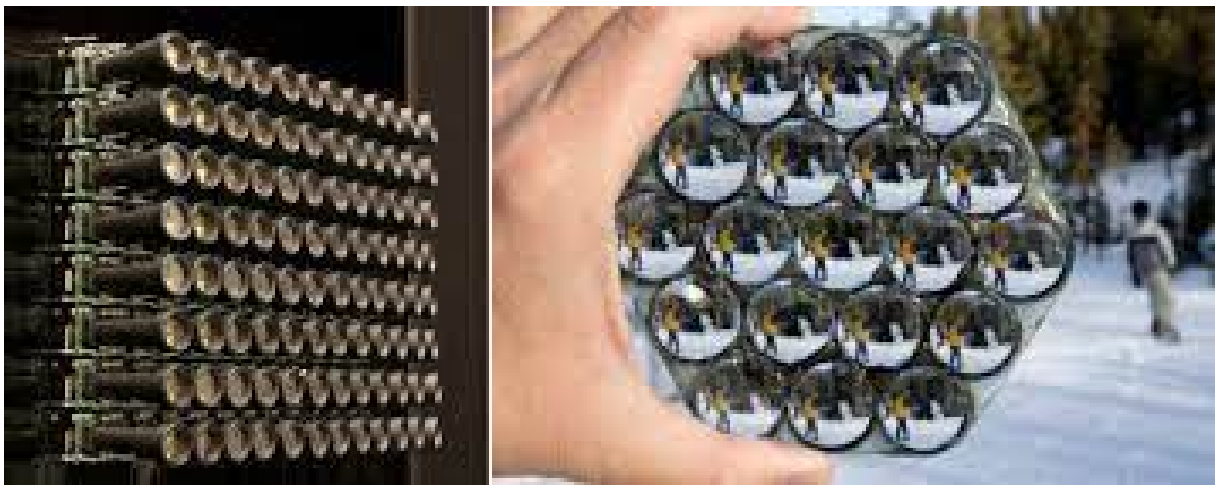

Figure 2.2: Light Field Acquisition: Camera array in the left, Microlenses in the right

2.3.2 Voxel Grid

Voxel grid representation is a specific type of 3D content representation especially in computer graphics. It involves dividing the 3D space into a regular grid of volumetric elements called voxels, similar to pixels but in 3D space. Each voxel in the grid stores information about the properties of the object or scene it represents, such as color, texture, density, or material properties. Voxel grid representation organizes the 3D space into a regular grid, where each voxel occupies a fixed and uniform volume. This structured arrangement makes it easier to access and manipulate voxels efficiently. Unlike mesh polygon representation that focuses on the surface of 3D objects, voxel grid representation encompasses the entire volume of the object or scene. This makes it well-suited for volumetric rendering and visualization. Voxel grid representation is commonly used in medical imaging, simulations, and scientific visualization, and also in virtual reality and 3D games which have been used to create destructible environments and realistic particle effects. But still, the voxel grid had some limitations like require a large amount of memory, especially for high-resolution grids or complex scenes, as each voxel needs to store its attributes. And Rendering and manipulating voxel grids in real-time can be computationally demanding, making optimization techniques crucial for interactive applications. Example of Voxel Grid is shown in Fig. 2.3 [8].

2.3.3 Polygon Meshes

Polygon Meshes are a widely recognized 3D representation in computer graphics, particularly in the realm of computer games. They form the surface or ”skin” of an object and are composed of a collection of polygons or faces. Representing a mesh model involves specifying points (vertices) in 3D space and connecting them with edges to create polygons. Typically,

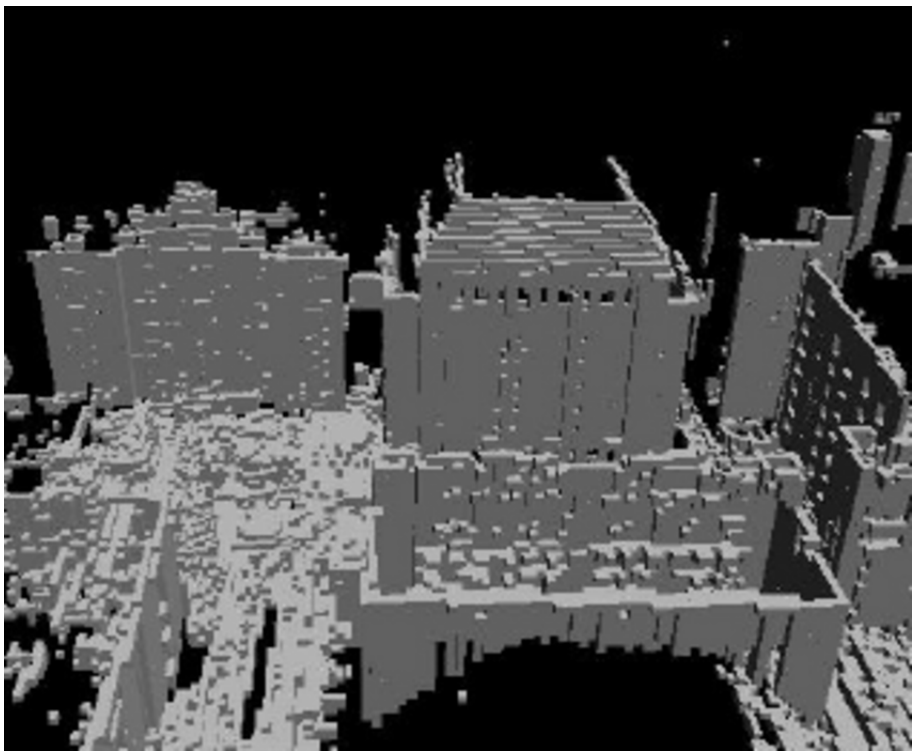

Figure 2.3: Voxel Grid Representation

three vertices combine to form one polygon, and multiple polygons together shape a surface, ultimately constructing the entire 3D object. However, polygon meshes have limitations. Representing smooth and curved surfaces can be challenging, requiring higher polygon density and increased memory and processing demands. Fine details like hair or flowing liquids may not be accurately represented with polygon meshes, necessitating the use of specialized techniques such as particle systems or advanced shaders. Despite these limitations, polygon meshes remain a powerful 3D representation, favored for their flexibility and efficiency in creating visually appealing and immersive virtual worlds. By combining polygon meshes with other techniques, computer graphics artists and game developers can bring life to captivating and interactive virtual environments. Example of Polygon Meshes is shown in Fig. 2.4.

2.3.4 Point Cloud

A point cloud is a collection of points in a 3D coordinate system, typically represented by XYZ axes. Each point in the cloud contributes to representing a spatial part of an object or scene. Additional attributes, such as color information and reflectance, can be assigned to points to convey specific characteristics. Point clouds can be categorized based on their characteristics, such as being static or dynamic, sparse or dense. In immersive content representation, dense and dynamic point clouds are often utilized to achieve more detailed and interactive virtual

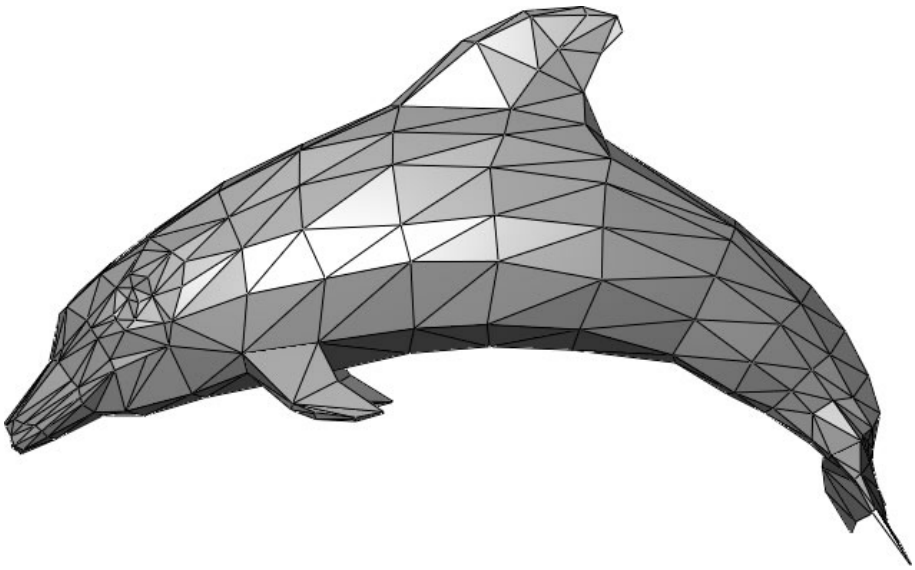

Figure 2.4: Polygon Meshes Representation

experiences. Point clouds can be generated through various methods, including 3D scanners, LiDAR technology, or photogrammetry software. The more points used to represent an object, the higher the level of detail and accuracy in the resulting representation. Point clouds play a crucial role in capturing the spatial information of real-world environments and objects, making them essential in various applications like virtual reality, computer graphics, and 3D modeling. Examples of Point cloud are shown in Fig. 2.5 [9].

Figure 2.5: Point Cloud Representation

2.3.5 Multiview plus Depth

The MVD format is a widely used 3D video representation in immersive media and various 3D video applications. The core idea behind MVD is to capture a scene using multiple cameras, each representing the scene from a different viewpoint, along with depth information for each captured view. By combining these multiple views and depth data, MVD aims to create a

more realistic and immersive visual experience. The process of creating an MVD involves capturing the scene from multiple cameras simultaneously, allowing for the acquisition of different perspectives. These multiple views enrich the content with depth cues, enhancing the sense of depth perception and ultimately delivering a more immersive visual experience to the viewer. The ”depth” component in MVD refers to the additional information about the distance from each camera to every pixel in the captured views. This depth data plays a crucial role in accurately reconstructing the 3D geometry of the scene. With the knowledge of depth, the system can determine the precise position of each pixel in 3D space, enabling the creation of stereoscopic effects and facilitating the rendering of virtual viewpoints between the original camera positions. MVD is commonly employed in various 3D video applications, virtual reality, and augmented reality systems. Its adoption is driven by its ability to provide a richer and more immersive visual experience for users. In this thesis, the focus will be on studying MVD format as an input format for the system. Example of MVD in Equirectangular Projection Form (ERP) is shown in Fig. 2.6 [10].

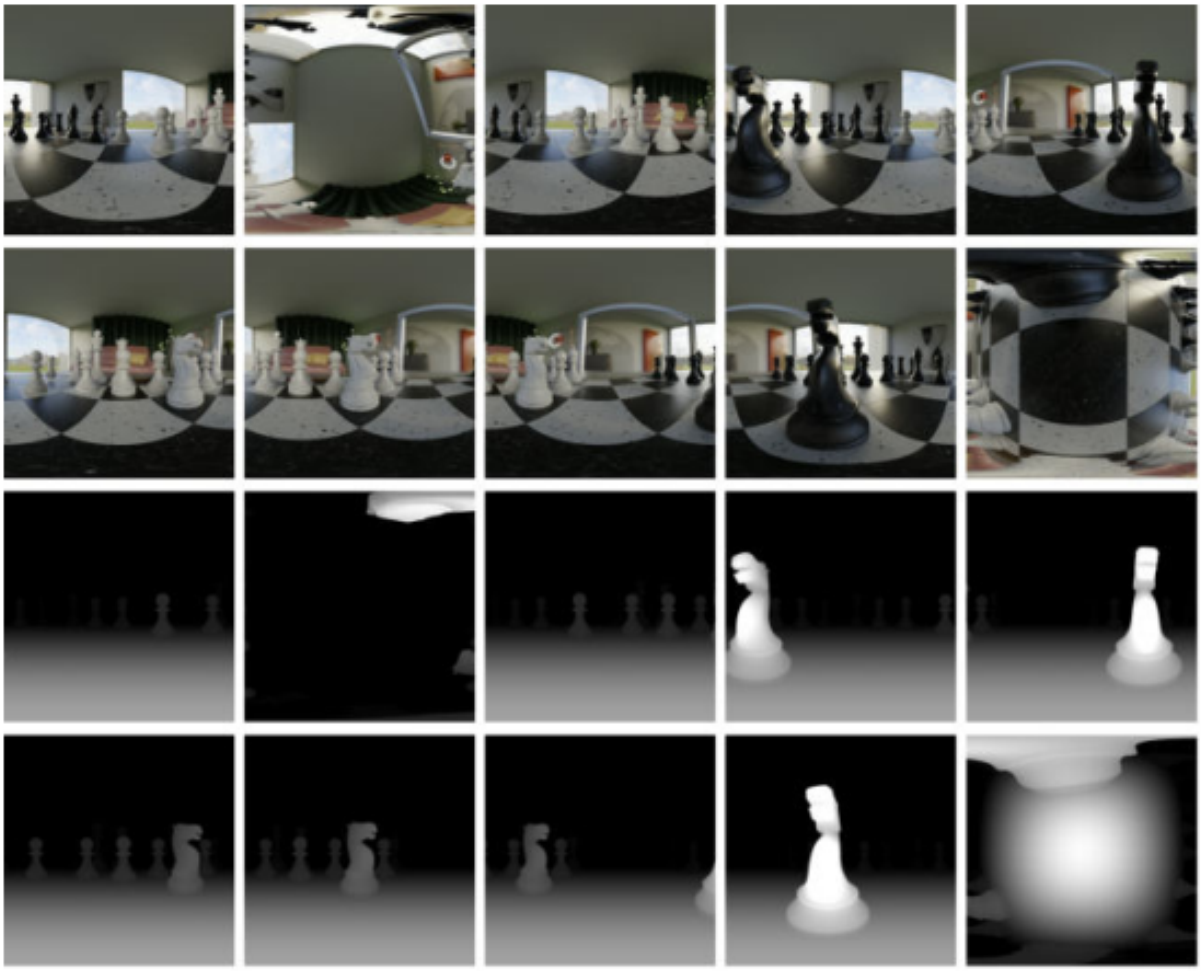

Figure 2.6: Mutltiview plus Depth Representation in Equirectangular Projection Form (ERP)

2.3.6 Multiplane Image

A Multi-Plane Image (MPI) is a representation of a 3D scene that involves multiple layers. The scene is divided into planar or spherical layers, each sampled at different depths from a specific reference point of view. Each layer is essentially a frame with color and transparency, obtained by projecting the part of the 3D scene surrounding the layer's location onto the same reference camera. The reference camera is positioned at the specified reference point of view and can be a perspective camera for planar layers or a spherical camera, typically equirectangular, for spherical layers. The MPI approach allows for the creation of a layered representation of the scene, enabling the portrayal of depth and perspective in a visually compelling manner. This technique is commonly used in computer graphics and virtual reality to enhance the realism and immersive experience of 3D scenes. Example of Multiplane Image representation in perspective form and equirectangular form is shown in Fig. 2.7 [11].

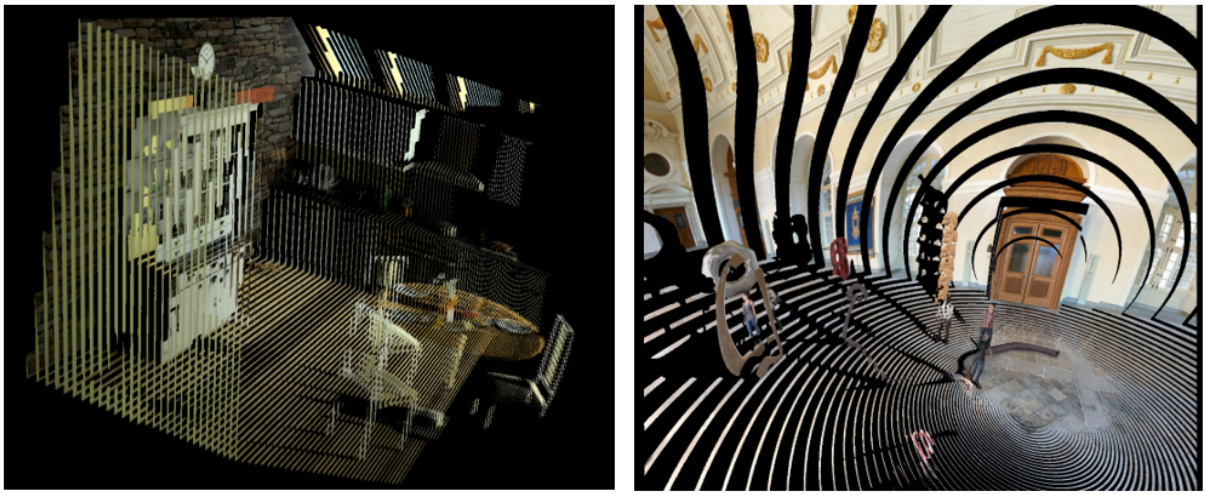

Figure 2.7: Multiplane Image Representation; Perspective Form (Kitchen in the left) and Equirectangular Form (Museum in the right)

2.4 Volumetric Video Coding

The MPEG group has developed two methods for compressing volumetric data: video-based compression and geometry-based compression. For video-based compression, they introduced the Video-based Point Cloud Compression (V-PCC) standard (ISO/IEC 23090-5) [12] and the MIV standard (ISO/IEC 23090-12) [13]. On the other hand, for geometry-based compression, the MPEG group established the Geometry-based Point Cloud Compression standard (ISO/IEC 23090-9) [14]. The following section will delve into a more detailed explanation of Video-based Point Cloud (V-PCC).

2.4.1 Video-based Compression

The two most renowned standards that employ video-based compression are Video-based Point Cloud (V-PCC), defined as part 5 of the MPEG group standards (ISO/IEC 23090-5), and MIV, defined as part 12 of the MPEG group standards (ISO/IEC 23090-12). In video-based compression, 2D video compression standards such as H.264 [15], H.265 [16], and H.266 [17] are utilized to compress and decompress the final results, which is the common part between V-PCC and MIV. However, the process of converting input views or input data to 2D videos and utilizing the common bitstream requires several steps. Both V-PCC and MIV need to generate patches and atlases, which are produced after removing redundancy from the input views.

The decision to utilize existing 2D video compression standards is based on several factors. Firstly, considerable time and effort have been invested in studying and refining these standards, making it a prudent choice to leverage established techniques for new standards and technologies. Additionally, 2D video decoders are widely prevalent in user devices, obviating the need for users to acquire new hardware for decoding the data.

As with all standards developed by the MPEG group, normative bitstreams and decoding instructions are defined, while leaving room for customization of the encoder and render parts by researchers and companies. MPEG offers valuable suggestions on how to approach the encoder and render aspects, exemplified by TMIV provided for the MIV standard. This model provides comprehensive guidance on compression and decompression processes, serving as a helpful reference for researchers. However, it is important to note that these guidelines are not rigidly enforced and can be adapted and refined as needed.

The next chapter will delve into further details about the MIV standard, elucidating the compression and decompression procedures from start to finish, providing more details on how to encode MVD to generate 2D videos bitstreams in the encoder path and how to reverse the operation in the decoder path until reaching the viewport that can be seen by the user through a head-mounted display. By combining established 2D video compression techniques with innovative technologies, the aim is to achieve efficient and high-quality representation of immersive content, providing users with compelling and immersive visual experiences. An example of a Video-based compression Pipeline for MIV standard on the encoder side using H.265 is shown in Fig. 2.8.

When compressing volumetric data presented in MVD format using 2D compression standards, for example, High Efficiency Video Coding (HEVC), several extensions have been proposed, including 3D High Efficiency Video Coding (3D-HEVC) [18]. However, 3D-HEVC

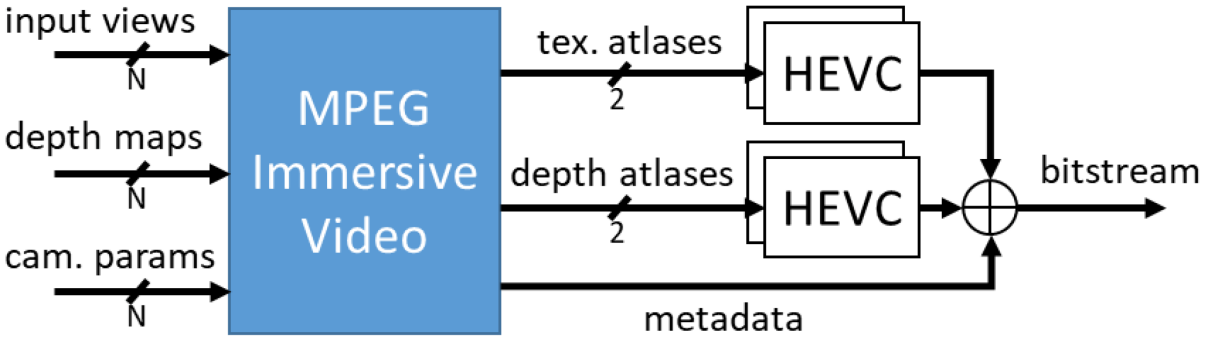

Figure 2.8: Video Based Compression (MIV encoder side)

comes with certain drawbacks. It introduces added complexity compared to traditional 2D video codecs like H.265/HEVC. This increased complexity results in longer processing times and higher computational requirements during the encoding and decoding processes of 3D video content. Furthermore, 3D-HEVC necessitates specialized hardware and software support for efficient encoding and decoding of 3D content. As a result, compatibility issues may arise with older devices or systems that lack 3D-HEVC capabilities, potentially requiring additional hardware upgrades or software installations. While 3D-HEVC offers benefits for volumetric data compression, its drawbacks should be carefully considered when choosing the appropriate compression method for specific applications. An example of a compression Pipeline using 3D-HEVC is shown in Fig. 2.9.

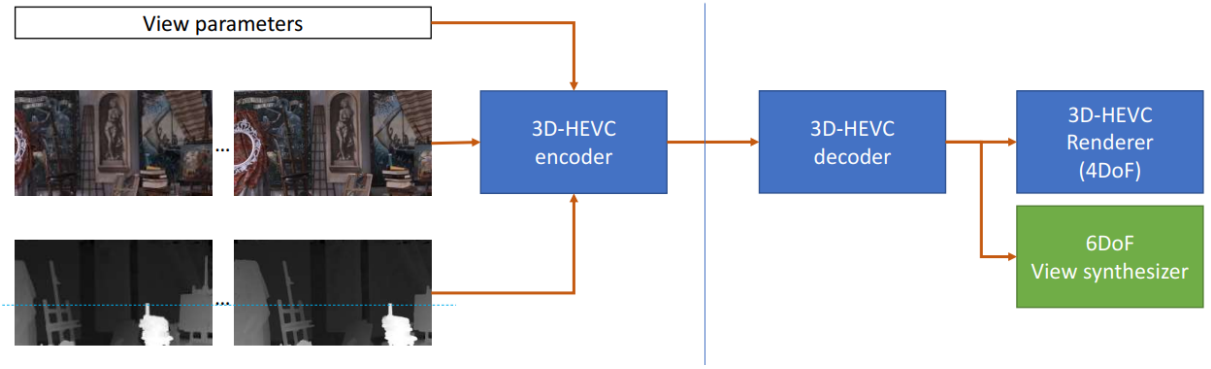

Figure 2.9: Pipeline for Compression using 3D-HEVC

2.5 Conclusion

In conclusion, the rise of new technologies like virtual reality, mixed reality, and augmented reality has ushered in an era of immersive media experiences. These advanced techniques have revolutionized content consumption by immersing users in virtual worlds with unparalleled realism and freedom. Immersive media offers a level of engagement and interaction that sur-

passes traditional methods, allowing users to explore virtual environments without borders or limitations.

Throughout this chapter, we have explored the concept and significance of immersive media, understanding its potential to transform user experiences. We delved into various methods used to implement immersive content, such as point cloud representation and MVD, each with its unique advantages in creating compelling virtual worlds. Furthermore, we explored several techniques used for compressing volumetric data, with a particular focus on video-based compression. During this exploration, we also observed how 2D video coding was ingeniously employed in the context of volumetric data compression.

In the next chapter, we will shift our focus to delve deeper into the MIV Standard as a state-of-the-art technique for volumetric video coding. We will explore TMIV, providing a comprehensive understanding of the entire pipeline—from the acquisition of input views to the final rendering stage.

Chapter 3

MPEG Immersive Video Standard

3.1 Introduction

The straightforward approach to encoding MVD sequences using the Multicast approach which involves assigning a separate encoder for each input view. Each texture and depth view is encoded using a separate bitstream, meaning that for each input view, there is one encoder for texture and one for geometry. The decoding process on the receiver side mirrors this, with one decoder for each bitstream to reconstruct the original input views. This encoding and decoding process utilizing High Effective Video Coding (HEVC) is illustrated in Figure 3.1.

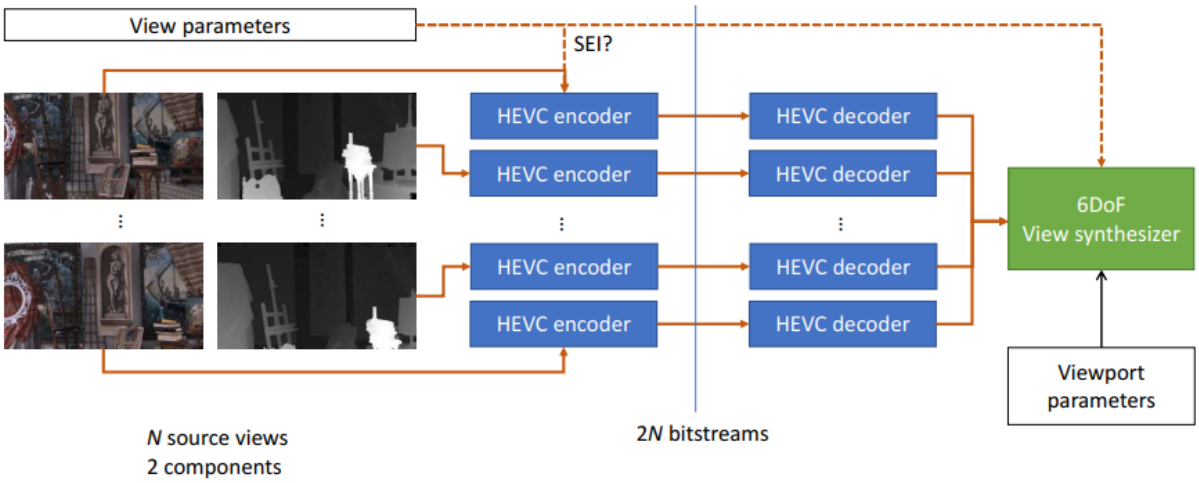

Figure 3.1: Pipeline for Compression using Multicast-HEVC

However, the Multicast approach is not considered very effective. Another approach is to use Multiview coding, which reduces the number of bitstreams to just two—one for texture inputs and one for geometry inputs. Despite this improvement, there are still redundancies present between the input views, leading to suboptimal utilization of the bitstreams. The encoding and decoding process utilizing High Efficiency Video Coding (HEVC) is illustrated in

Figure 3.2.

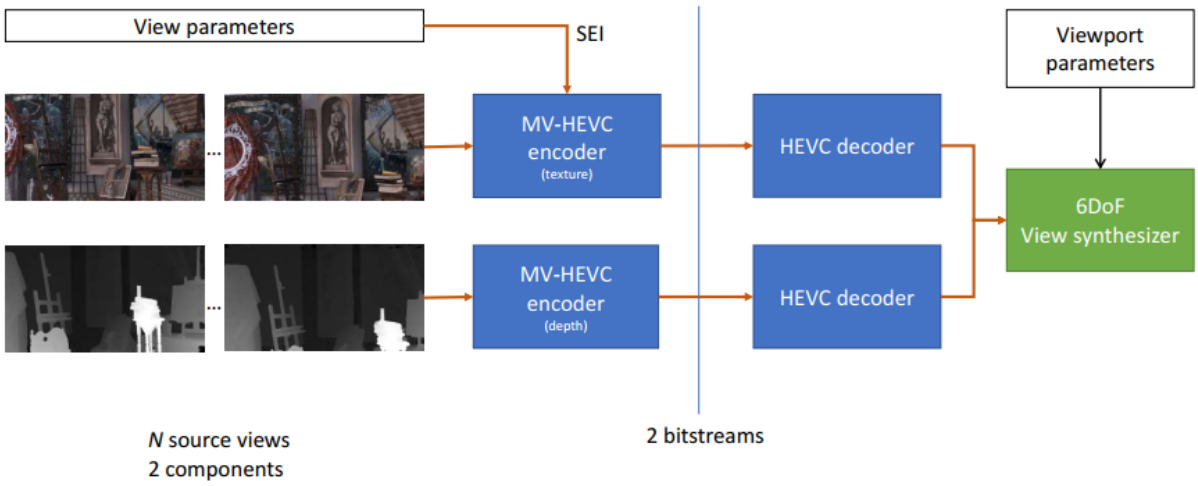

Figure 3.2: Pipeline for Compression using Multiview-HEVC

To overcome the limitations of Multicast and Multiview coding, MIV standard was introduced by the MPEG group, which is a part of the ISO/IEC MPEG-I family of standards (part 12). The primary objective of MIV is to facilitate the compression of immersive video content, wherein real or virtual 3D scenes are captured using multiple cameras, either real or virtual. This standard empowers the storage and distribution of immersive video content over both existing and future networks, allowing playback with six degrees of freedom for view position and orientation. With MIV, users can experience immersive videos with a high degree of interactivity and freedom, enhancing the overall viewing experience in virtual and augmented reality environments. Figure 3.3 show MIV codec model in a general way [19].

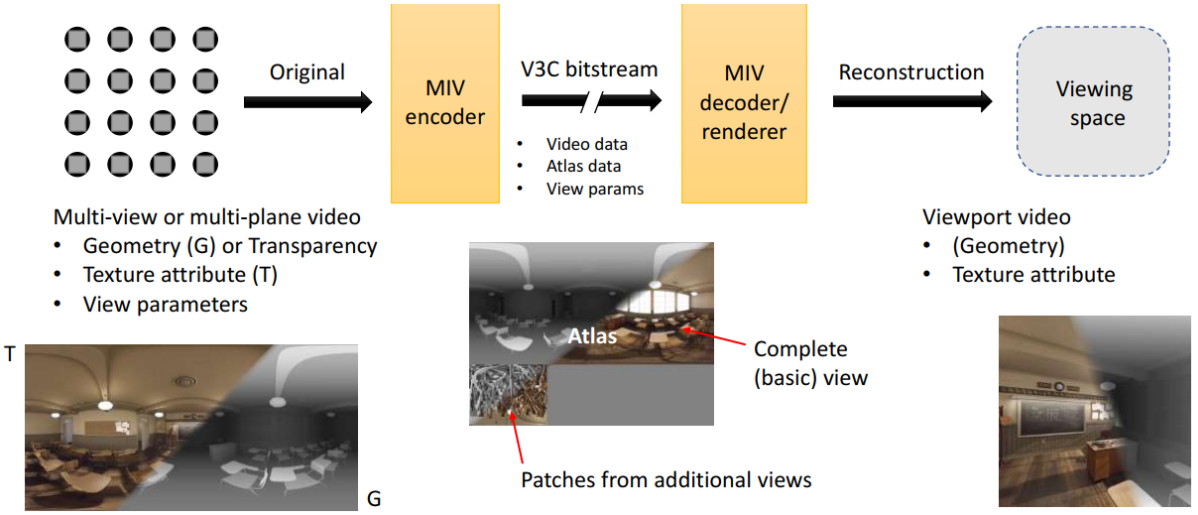

Figure 3.3: MIV Codec Model

As with any standard proposed by MPEG, MIV defines a normative bitstream and decoding processes. To demonstrate the implementation of the MIV standard, a reference model called

TMIV was provided.

In the following section, I will provide a more detailed explanation of how TMIV works and how it overcomes these challenges.

3.2 Test Model MPEG Immversive Video

TMIV introduces a novel technique called pruning, which aims to enhance bitstream efficiency by removing redundancy between input views that is not crucial for rendering in the decoder.

During the pruning process, certain groups of pixels, known as patches, are identified and packed together to form Atlases. The Atlases serve as the output format of the TMIV encoder, containing both full views from basic views and patches from additional views. As a result of pruning; one Atlas will include full views and one Atlas will include patches.

After the Atlas operation, the Atlases are encoded using 2D video codec techniques and packed into a bitstream along with related information. This process is performed for both texture views and geometry views. The goal of doing pruning is to respect the pixel rate constraint. Pixel rate constraint in compression refers to a limitation on the number of pixels that can be processed or transmitted per unit of time during the compression process. Alternatively pixel rate can be defined as "The number of luma pixels that have been decoded per second in order to play a video in real-time". In Figure 3.4, an example of texture and depth Atlases is illustrated [10].

3.2.1 MIV Profiles

Similar to other MPEG codecs, MIV offers multiple use cases to cater to diverse groups of users with varying requirements. These use cases are organized and defined using profiles. Each profile within MIV consists of a set of features that are specifically enabled by the specification to target distinct application classes. By providing different profiles, MIV ensures that it can be flexibly adapted to meet the specific needs and preferences of different users and applications. MIV offers three primary profiles: MIV Main Profile, MIV Extended Profile, and Geometry Absent Profile.

The MIV Main Profile is the first profile offered by MIV. It focuses on providing the essential features required for virtual reality applications. This profile is particularly well-suited for applications that utilize MVD as the input format. However, it is important to note that this profile does not support separate occupancy map, and both texture atlases and geometry are

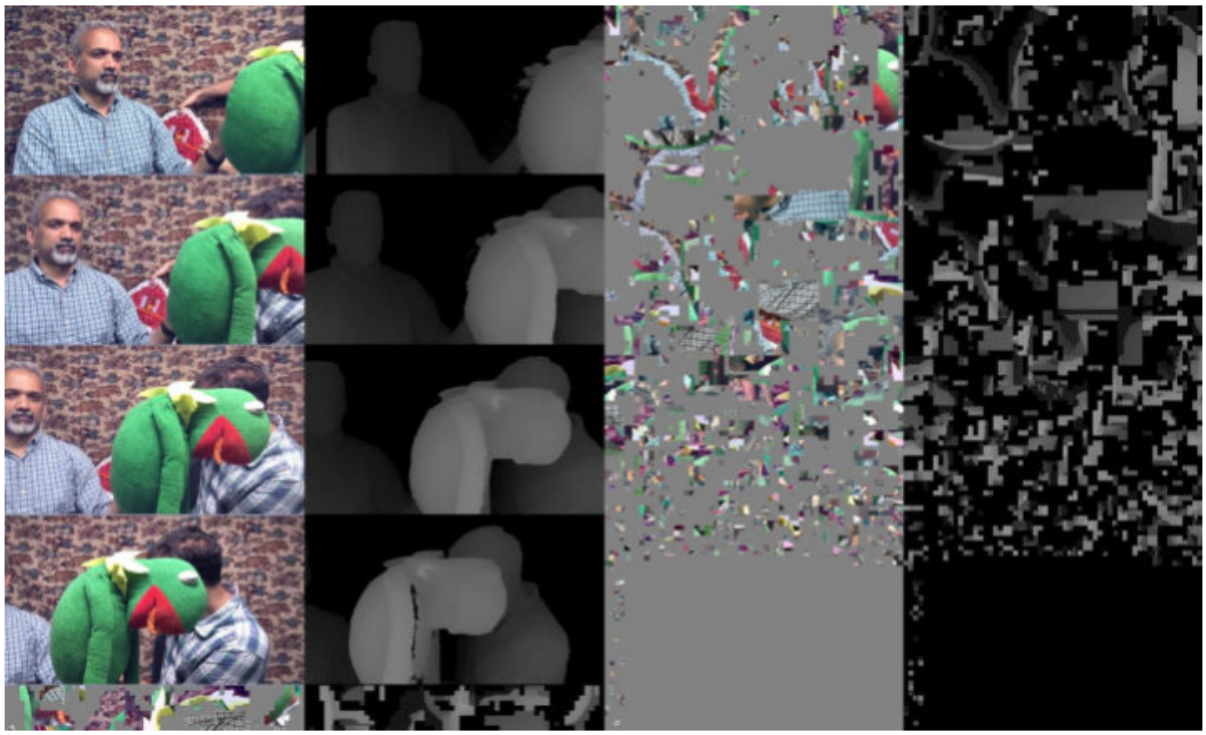

Figure 3.4: An example of atlases for texture and depth, first column: texture Atlas with four basic views and patches, second column: depth Atlas with four basic views and patches, third column: texture Atlas with texture patches, fourth column: depth Atlas with depth patches

encoded independently.

The MIV Extended Profile is the second profile offered by MIV. This profile builds upon the features provided by the MIV Main Profile by introducing additional tools, such as support for external occupancy maps. Occupancy maps are additional videos that contain binary data, indicating whether a specific pixel in the related geometry or texture videos belongs to a valid point in the scene. By enabling support for external occupancy maps, this profile enhances the efficiency of the encoding process. Moreover, the MIV Extended Profile allows texture atlas and geometry atlas to be packed together, rather than being encoded independently as in the MIV Main Profile. This optimization reduces the number of decoders required in the decoding process. Additionally, this profile includes a sub-profile known as the Restricted Geometry Profile. The Restricted Geometry Profile is commonly used for applications that utilize Multiplane and Multishere (MPI/MSI) as inputs. In the Restricted Geometry Profile, only texture and transparency attributes are coded as video sub-bitstreams. This profile is well-suited for real-time rendering on low-end devices due to its less complex rendering algorithm. Overall, the MIV Extended Profile expands upon the capabilities of the MIV Main Profile, providing more advanced tools and optimizations for immersive video coding.

The Geometry Absent Profile is the third profile offered by MIV. In this profile, the encoder

focuses solely on encoding the texture atlases, without including geometry atlases. The depth information is then estimated on the client side using a decoder side depth estimation technique. By adopting decoder side depth estimation, the computational burden is shifted from the encoder to the decoder. This profile is well-suited for applications where the decoders are computationally powerful and capable of performing real-time depth estimation. By offloading the depth estimation process to the decoder side, the Geometry Absent Profile allows for more efficient compression and transmission of immersive video content. However, it does require the decoder to have sufficient processing capabilities to handle the depth estimation task effectively. Overall, the Geometry Absent Profile provides a trade-off between computational complexity at the encoder and decoder sides, making it a suitable choice for certain applications with appropriately equipped decoders [20]. Figure 3.4 shows MIV profiles with possible inputs format.

	Profiles			
Name	Main	Extended		Geometry Absent
			Restricted subprofile	
Video Bitstreams	G, T	G, T, A, O	T, A	T
Usage	MVD	MVD+	MPI	Depth generation by client

Legend: G geometry, T texture, A Transparency, O Occupancy sub-bitstreams

Figure 3.5: MIV Profiles

3.2.2 Encoder

The primary objective of the TMIV Encoder is to create one or more attribute and geometry atlases by compositing patches extracted from input views. Additionally, the TMIV Encoder generates metadata that describes the composition of the atlases. The geometry and attribute atlases are then encoded as videos using a 2D video encoder, while the metadata is encoded in the bitstream using the MIV standard.

Similarly, like the Visual Volumetric Video-based Coding (V3C) approach, MIV can be used with any video coding standard. During the development process, HEVC was predominantly

utilized since it was widely deployed in various products. However, it is worth noting that V3C explicitly defines codec profiles, enabling support for other video coding standards such as Advanced Video Coding H.264 (AVC) and Versatile Video Coding H.266 (VVC), and provides a mechanism to indicate the use of alternative codecs. This flexibility allows MIV to adapt to different scenarios and requirements [13]. Figure 3.6 show a high-level block diagram of MIV encoder.

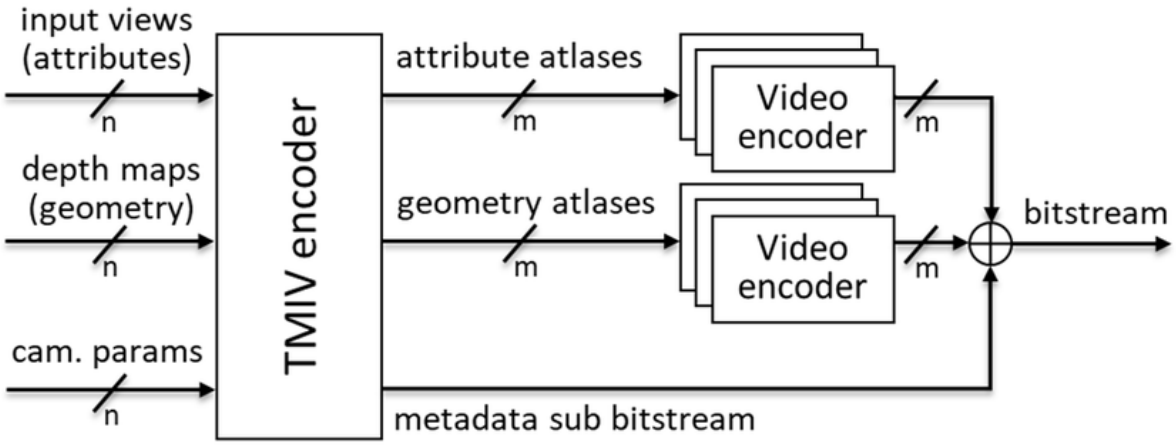

Figure 3.6: High-level block diagram for the MIV encoder

The TMIV encoder employs a series of steps to effectively compress the volumetric data. It begins with Geometry Quality Assessment, ensuring the quality of the geometry data. Optionally, the input views can be split into groups, then views labeled as basic or additional. Basic views are encoded as full views to generate an atlas, while additional views are pruned to produce patches, resulting in one or more atlases. Each group is then encoded separately.

The attribute and geometry atlases are separately encoded as videos using the HM-HEVC reference model. Finally, the coded video bitstreams, along with the metadata sub-bitstream, are combined to create the final MIV bitstream. This high-level scheme is depicted in Figure 3.7 for the TMIV group-based encoder.

Within each group, the encoding process undergoes various stages. It starts with automatic parameter selection, then optionally, the entity layers can be separated. The encoder then utilizes the prune graph to remove unnecessary pixels, followed by aggregating pruning masks and clustering active pixels. Clusters are split, and patches are packed to generate the video data [13]. This high-level scheme is illustrated in Figure 3.8 for the TMIV single-group encoder.

In the following subsections, I will provide a brief explanation of each stage, clarifying its purpose and function in the encoding process.

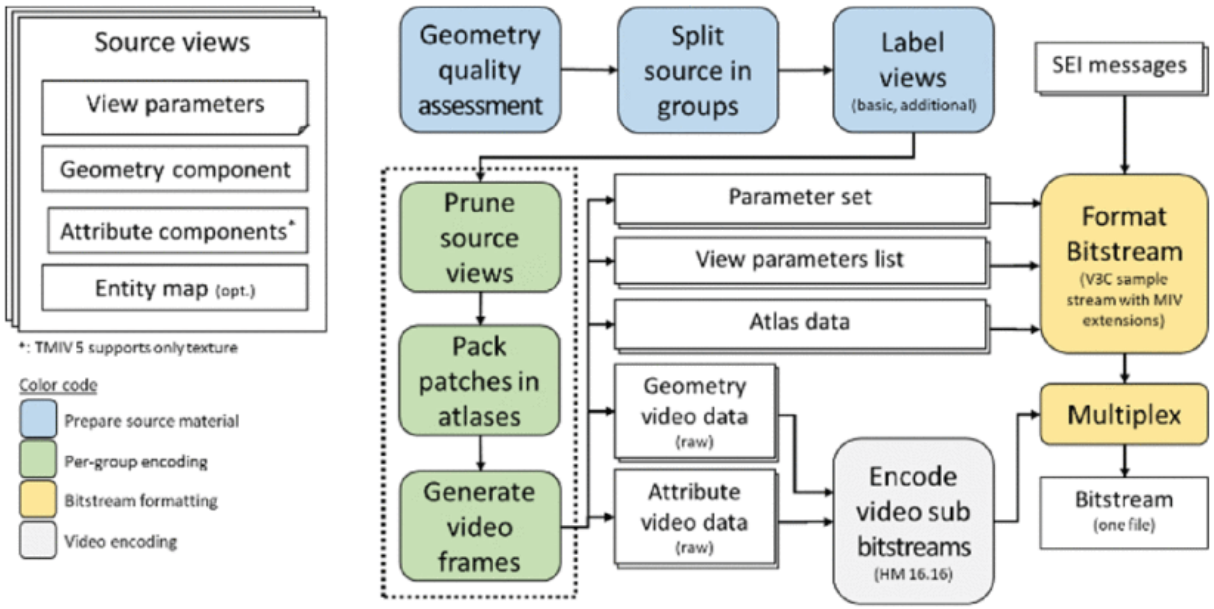

Figure 3.7: High-level scheme of the TMIV group-based encoder

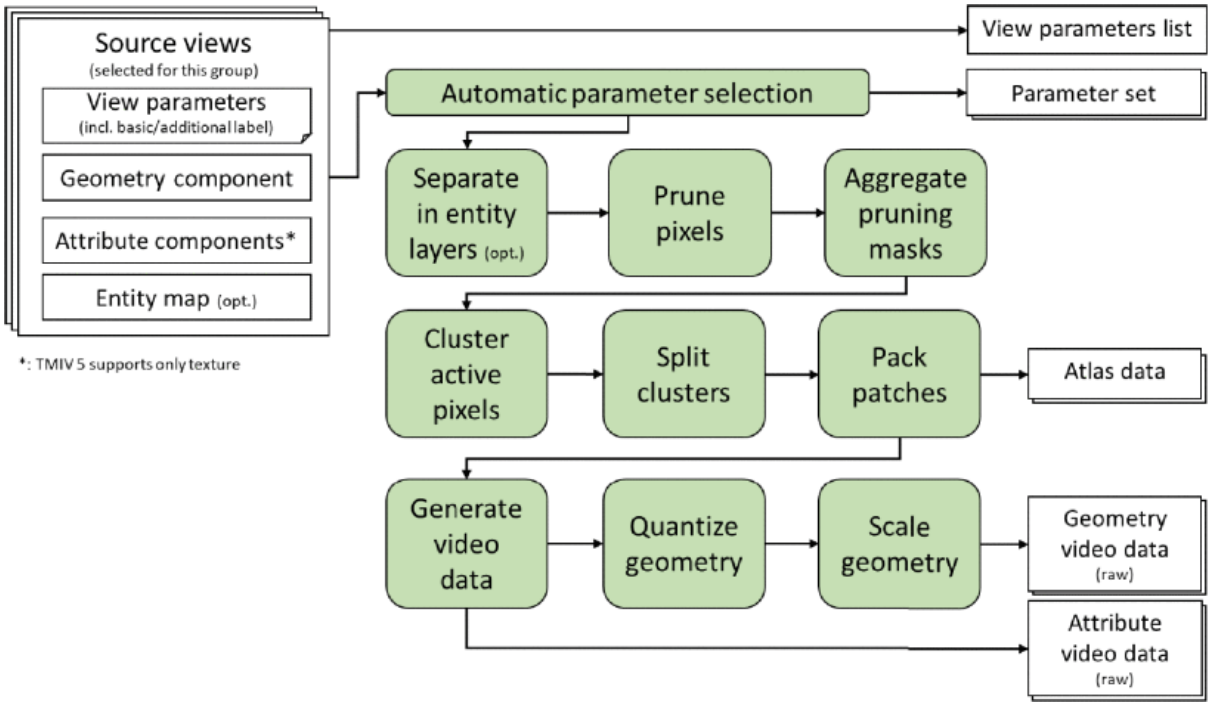

Figure 3.8: High-level scheme of the TMIV single-group encoder

3.2.2.1 Encoder Inputs

The TMIV encoder takes source views as input, representing the projection of the 3D real or virtual world. When working with the MVD input format, these source views consist of various components, including the texture attribute components, geometry components (in the case of the MIV main profile), and view parameters.

View parameters provide crucial information about the characteristics of the cameras used to capture the scene. This includes intrinsic parameters like focal length and principal point for perspective projection, extrinsic parameters like camera position and orientation, projection type (e.g., perspective, equirectangular, orthographic), projection plane size, and depth quantization parameters (QPs).

In addition to the main components, the source views may also contain an entity map. The entity map is a binary mask that indicates the presence or absence of entities (objects or elements) in the scene for a specific view. It helps in identifying areas of interest and can be used to split views based on entity maps during the encoding process. An example of source views with their components, including texture attributes, geometry, and entity maps, is shown in Figure 3.9 [11].

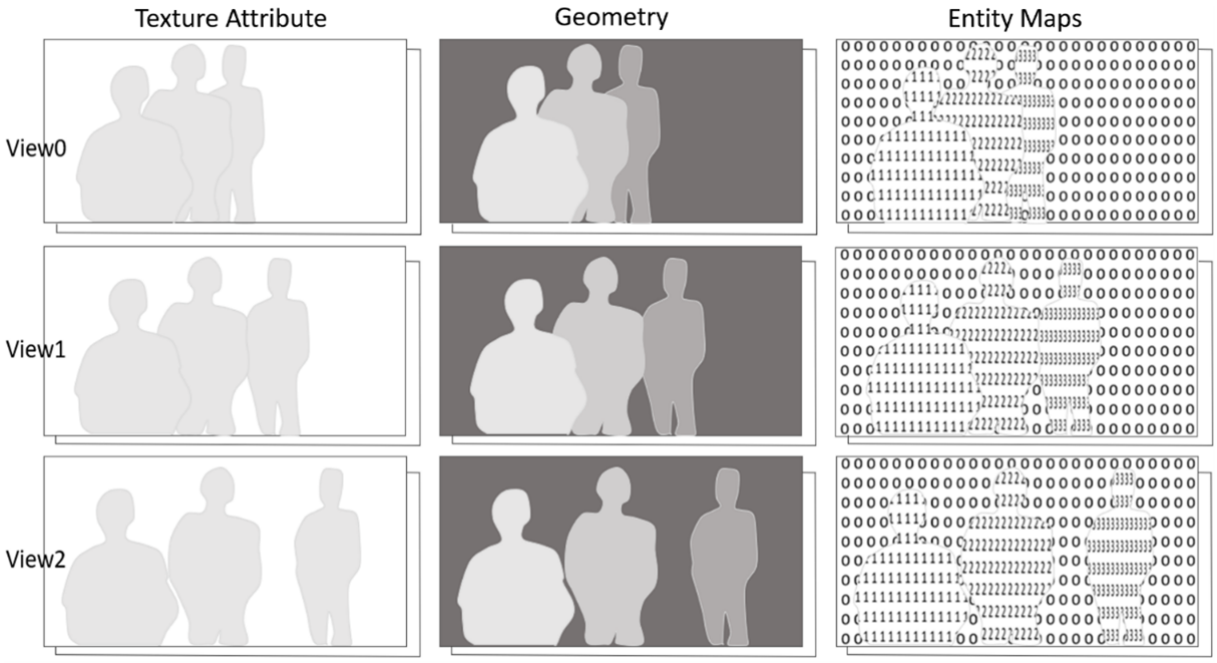

Figure 3.9: source views with geometry, textures, and entity maps

3.2.2.2 View Preparation

Once the encoder receives the input source views, several preparatory operations are performed before spatially encoding each group. The first crucial step is splitting the source views

into groups and then labeling the views as either ”basic” or ”additional” views.

The encoder **distributes the source views into groups**, although this step is optional and depends on the number of source views available. The grouping process aims to identify and focus on local coherent projections of important regions within the scene. This becomes particularly advantageous in multicamera rig systems, where distant views might have less in common and can be processed more efficiently in separate groups while still being multiplexed in the final bitstream.

The selection of views for each group is done automatically and depends on the total number of groups and source views. Each group is encoded independently of the others, allowing for parallel processing, which can significantly enhance encoding efficiency. By grouping views that share commonality and discarding those with little to no commonality, unnecessary processing of distant views is avoided.

The number of groups can be defined by the user, offering flexibility in adapting the encoding process to the specific characteristics of the input data. An example of how grouping works is shown in Figure 3.10, where the source views are divided into four groups. Each group is then encoded separately, and finally, the bitstreams from all groups are merged in the last stages of the encoding process to form the final MIV bitstream [11].

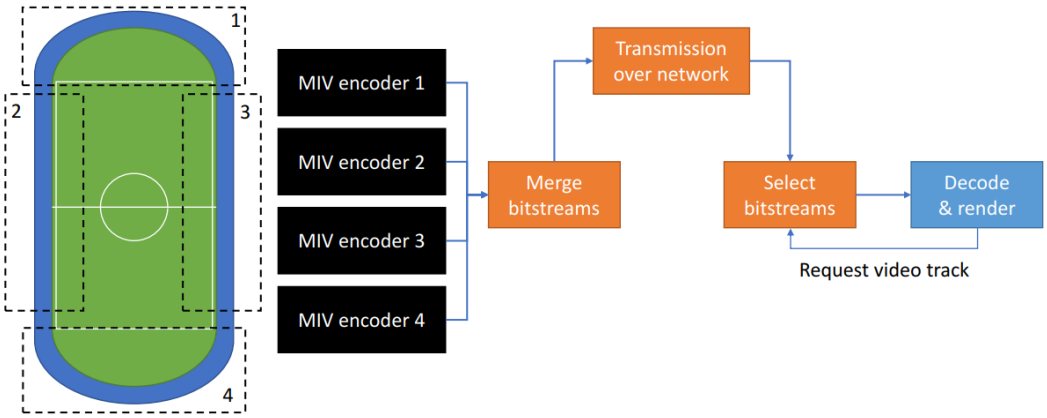

Figure 3.10: Group-based encoding

After splitting the source views into groups, the next step is **view labeling**, which is performed in two stages: view selection and basic view allocation.

During **view selection**, two operation modes are available. In the first mode, all source views are output after labeling, and they are classified as either basic or additional views. The output will consist of one or more atlases containing complete views and one or more atlases containing patches. Figure 3.11 illustrates an example of this mode.

In the second operation mode, only basic views are included in the output. The result will

Figure 3.11: view selection with additional views

be one or more atlases containing only complete views. Figure 3.12 provides an example of this mode. The choice of operation mode depends on the specific requirements and characteristics of the application.

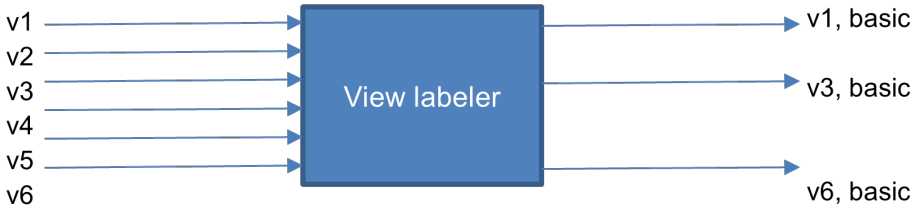

Figure 3.12: view selection without additional views

After the view selection operation, the next stage is **basic view allocation**. This step involves determining the number of basic views based on constraints such as the number of atlases per group and the luma picture size. Once the number of basic views is calculated, they are selected based on their distance and overlap with other source views.

3.2.2.3 Automatic Parameter Selection

In the single-group encoding process, the first step is automatic parameter selection, which involves two operations. The initial operation is geometry quality assessment, where the quality of the geometry maps is evaluated. This step is crucial as it affects subsequent processes, particularly pruning and rendering phases. The second operation in automatic parameter selection is atlas frame size computation which will compute the size of atlas frames.

During the **geometry quality assessment** process, this operation is done based on the first frame, each input view is reprojected to the position of all other remaining input views. For every reprojected pixel, the encoder checks whether the reprojected geometry value is higher than a specified threshold of the geometry value of the corresponding pixel or any of its neighbors in the target view (within a 3×3 neighborhood). If this condition is not satisfied, the pixel is deemed inconsistent. If the number of inconsistent pixels between any pair of input views exceeds a certain threshold, the geometry quality is considered to be low.

After evaluating the quality of the geometry maps, the TMIV encoder automatically **calculates the size of the atlas frames**. This calculation involves considering three constraints: the maximum size of the picture in terms of luma samples, the maximum sample rate of the luma samples in hertz, and the total number of allowed decoder instances. Based on these constraints, the encoder determines the dimensions of the atlas frame. The width of the atlas frame is set to accommodate the widest input source views, while the height is set as large as possible while still adhering to the aforementioned constraints. Additionally, the number of atlases per group is set high enough to achieve the maximum luma sample rate without exceeding it [13].

3.2.2.4 Separation in Entity Layers

In TMIV, there is an optional mode that operates based on entity maps, where entities in the scene can be objects or components. To utilize this mode, entity maps are provided in the source views, which assign a specific value to each pixel in the scene, indicating its association with a particular entity. If the entity mode is used, the encoder will perform the subsequent operations based on this mode. For example, pruning, aggregation, clustering, and packing will be executed considering the information provided by the entity maps. This approach allows the encoder to take advantage of entity-level information to optimize the compression process.

3.2.2.5 Pruning of Redundant Pixels

Pixel pruning is a critical step in the atlas construction process. Since the input source views capture the scene from different viewpoints, there is often a significant amount of redundancy among these views. To ensure efficient utilization of the bitstream, the pruning stage is employed to select which pixels can be pruned without compromising the rendering quality and which pixels should be preserved. The pruner operates at the frame level and across all views. It takes the texture attributes, geometry maps, and camera parameters as input and generates a new image, referred to as a ”mask,” for each view and frame. The mask has the same dimensions as the input views.

For basic views, the output mask remains the same as the input, indicating that no pruning is performed for basic views, and all pixels are preserved. However, for additional views, the output mask contains pixels that are non-redundant with other views, which will be preserved, while the pixels that are redundant with other views, will be pruned. By selectively pruning

redundant pixels in additional views, the encoder can significantly reduce redundancy and optimize the bitstream representation without compromising the visual quality of the rendered immersive content.

The pruner utilizes a **pruning graph** to establish a hierarchy for view pruning. The pruning graph outlines the relationships between different views, guiding the process of determining which pixels to prune in additional views while preserving essential information. An example of a pruning graph is illustrated in Figure 3.13, which includes one basic view and three additional views.

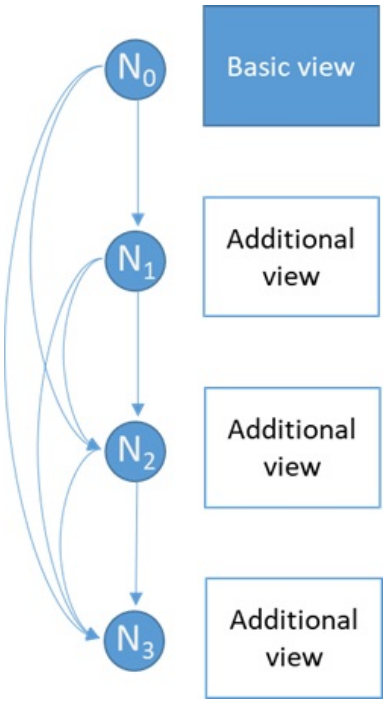

Figure 3.13: Pruning graph for one basic and three additional views

To construct pruning graphs, several steps must be followed. Firstly, the basic views are added to the pruning graph and assigned as root nodes. Next, all pixels of the basic views are projected onto each additional view. For each additional view, a pruning mask is created to identify pixels that are redundant with other views. The additional view that preserves the highest number of pixels is selected and added as a child node after the basic views in the pruning graph. This process continues until all views are assigned to nodes in the pruning graph.

Once an additional view is selected and added to the graph, the preserved pixels of this view are projected onto the remaining additional views, and the pruning masks for these views are updated accordingly. The process is then repeated for each view until all views are included in the pruning graph. By following these steps, the encoder can effectively identify and prune redundant pixels in the additional views while preserving essential information.

The pruner uses three criteria to determine whether a pixel should be pruned or preserved. Firstly, the pixel should be synthesized from views higher up in the hierarchy, which means it should be preserved in the parent node and pruned from the child node. Secondly, the difference between the synthesized geometry and the source geometry should be below a certain threshold. The third criterion involves considering the minimum difference in luma between the synthesized pixel and all the pixels within a collocated source 3x3 block, which should be below a pruning luma threshold.

In a second pass process, global color differences are taken into account. Pixel-by-pixel color differences are calculated between the synthesized view from the parent node and the source views assigned to the child node, using the least squares method. Pixels that comply with this fitting function within a certain range defined by a threshold are considered inliers and are pruned. On the other hand, pixels that fall outside this range are updated as not to be pruned in the pruning mask [11].

3.2.2.6 Pixel Clustering Into Patches

After the pruning process, the resulting masks are aggregated over an intra period. During this aggregation, the contours become thicker in the regions of the geometry maps where there is motion or significant changes. Figure 3.14 demonstrates the outcome of mask aggregation in an intra-period.

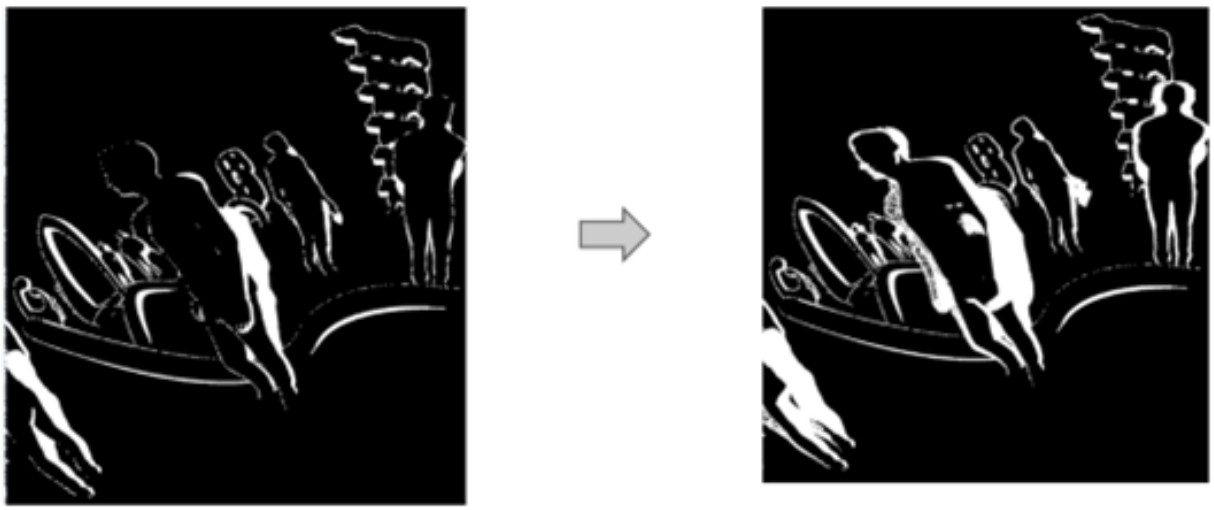

Figure 3.14: Mask aggregation within intra-period

In the next step, clusters are created from the aggregated masks. A cluster can be defined as a set of connected pixels on the aggregated mask. To determine the connection between pixels, we examine the eight neighbors surrounding active pixel. If at least one other pixel among the

eight neighbors is present, the pixel is considered connected and added to the cluster. Figure 3.15 show connectivity criteria with eight pixels neighborhood.

Figure 3.15: Connectivity with eight pixels neighborhood

Patches are formed by enclosing clusters with rectangular bounding boxes. These patches are defined by their x and y positions of the top left corner, as well as the height and width of the bounding box. Figure 3.16 illustrates an example of the patches and clusters, with clusters being visually highlighted in different colors.

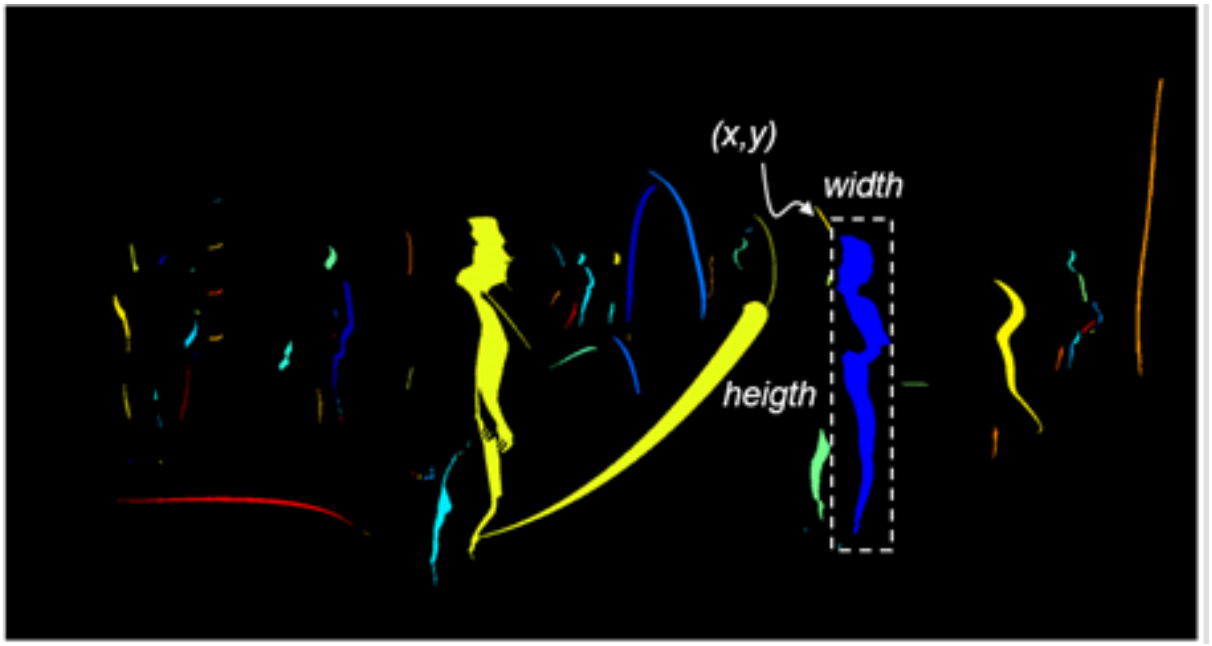

Figure 3.16: Patches and Clusters representation

To efficiently utilize the available atlas space, some patches, particularly those with irregular shapes, may occupy a large area within a rectangular box. To address this, cluster splitting can be performed, where each cluster is split into two smaller clusters. However, for splitting to occur, a condition must be met: the total area of the bounding boxes of the smaller clusters must be smaller than the bounding box of the initial cluster. The splitting is carried out along a line that is parallel to the shorter side of the bounding box, resulting in L-shaped clusters. Figure 3.17 provides an example illustrating the process of cluster splitting.

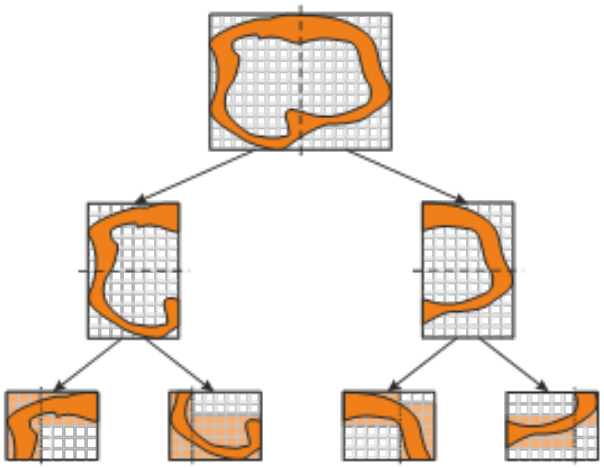

Figure 3.17: Cluster splitting

3.2.2.7 Patch Packing

During this phase, the patches will be packed into atlases in decreasing size order, with larger patches packed first followed by smaller patches. The packing process utilizes the MaxRect algorithm, which efficiently packs the patches within the available atlas space. Additionally, vertical flipping and rotations of 90, 180, and 270 degrees are supported to optimize the packing arrangement further.

To ensure reversibility in the decoder side, various information related to the packing process needs to be stored in metadata. This includes the position, rotation, patching order, and source view number for each patch. This metadata will allow the decoder to unpack the patches and reconstruct the original scene accurately.

Figure 3.18 provides an example of atlas generation using three source input views, using the kitchen sequences, illustrating how the patches are efficiently packed into the atlases.

3.2.2.8 Geometry Coding

The output of the previous stages is the generation of geometry and attribute atlases. These atlases need to be encoded using video coding techniques, with the number of bits determined by the specific MIV profile being used. While MIV does not prescribe a specific video coding standard, in common test conditions (CTCs), the HEVC Main 10 profile has been commonly used with a bit depth of 10 bits to encode both attribute and geometry atlases.

To ensure minimal quantization error in the geometry atlases, the quantization law is based on the normalized disparity for perspective and equirectangular projection types. This requires the transport of metadata that indicates the near and far limits (d_{near}, d_{far}), which correspond to the (Z_{min}, Z_{max}) range for each view [13]. The formula for quantization in the case of 10-bit

Figure 3.18: Input views in the left, views after pruning in the middle, atlas in the right

depth is expressed in the following Formula:

$$d_{quantized} = 1023 \frac{d - d_{far}}{d_{near} - d_{far}} \tag{3.1}$$

3.2.2.9 Geometry Downscaling

MIV employs downscaling in the geometry atlases to reduce the pixel rate. This involves decreasing the resolution of geometry frames while also lowering the quantization point (QP) of the HEVC encoder. By doing so, it becomes possible to use fewer pixels while maintaining a similar end-to-end rate-distortion (RD) characteristic.

To achieve downscaling, TMIV utilizes the max pooling operation. Max pooling is a technique commonly used in computer vision and image processing to reduce the size of feature maps while retaining essential information.

3.2.3 MIV Bitstream

The MIV standard provides a normative bitstream with a defined decoding process. The MIV bitstream is an extension of the Visual Volumetric Video-based Coding (V3C) bitstream format, which is commonly used in Video-based Point Cloud Compression (V-PCC) as part of the MPEG group standards (ISO/IEC 23090-5).

The V3C bitstream is composed of small units called V3C units, each containing a header and payload. The header's first element specifies the unit type. Various types of V3C units,

are defined in the standard, such as V3C parameter set (VPS), Atlas data (AD), common atlas data (CAD), geometry video data (GVD), attribute video data (AVD), occupancy video data (OVD), and packed video data (PVD). Each unit serves a specific purpose, with the VPS being essential as it signals the start of a new sequence and provides information about atlas count, atlas frame sizes, presence of unit types, and more.

V3C units can contain video sub bitstreams, including geometry video data (if present), attribute video data (AVD), occupancy video data (if present), and packed video data (if present) for each atlas component. They can also have atlas sub bitstreams, such as Atlas data and common atlas data.

TMIV, supports Packed video data V3C units, typically used when various video data, such as geometry with attribute atlases, are combined.

The Atlas data V3C unit contains an Atlas sub bitstream, which includes a network abstraction layer (NAL) unit stream. NAL has atlas tile layers (ATL). These ATLs carry a list of patch data units (PDU) that establish the relationship between patches in the atlases and the source views.

Similarly, the Common atlas data V3C unit contains an Atlas sub-bitstream with a NAL unit carrying the common atlas frame (CAF) that contains the view parameters list. There are also NAL units that carry Supplemental Enhancement Information (SEI) messages.

In summary, the MIV bitstream is organized with various V3C units, each serving specific functions and containing relevant video and atlas sub-bitstreams, forming the structure of the data for efficient decoding and rendering processes. Figure 3.19 illustrates the structure of the MIV bitstream [11] [13].

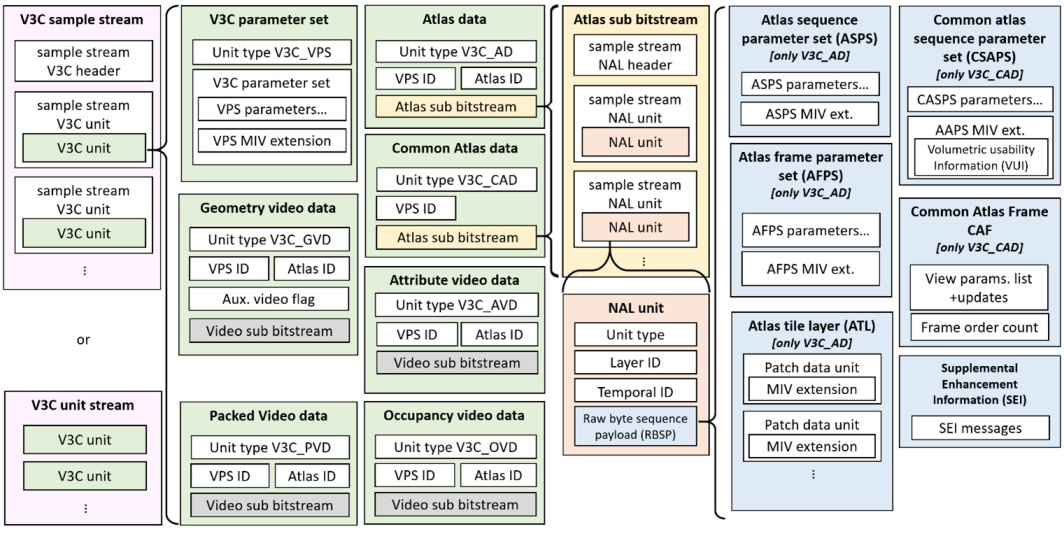

Figure 3.19: Structure of the V3C bitstream with MIV extensions

3.2.4 MIV Rendering Process

As mentioned earlier, the MIV standard provides a normative bitstream and decoding process, leaving the encoding and rendering aspects for researchers to explore and innovate. However, to illustrate the entire process from handling input source views in the encoder to the rendering process and delivering the desired viewport to users, a test model called TMIV has been proposed. Now, let's delve into how the rendering process is handled as proposed in TMIV.

The TMIV decoder adheres to the MIV decoding process outlined in the MIV specification, which includes various steps such as demultiplexing, decoding order, bitstream parsing, video decoding, frame unpacking, block to patch map decoding, and resulting conformance points. In the upcoming subsections, we will describe the non-normative rendering process, which starts from these conformance points, as depicted in figure 3.20.

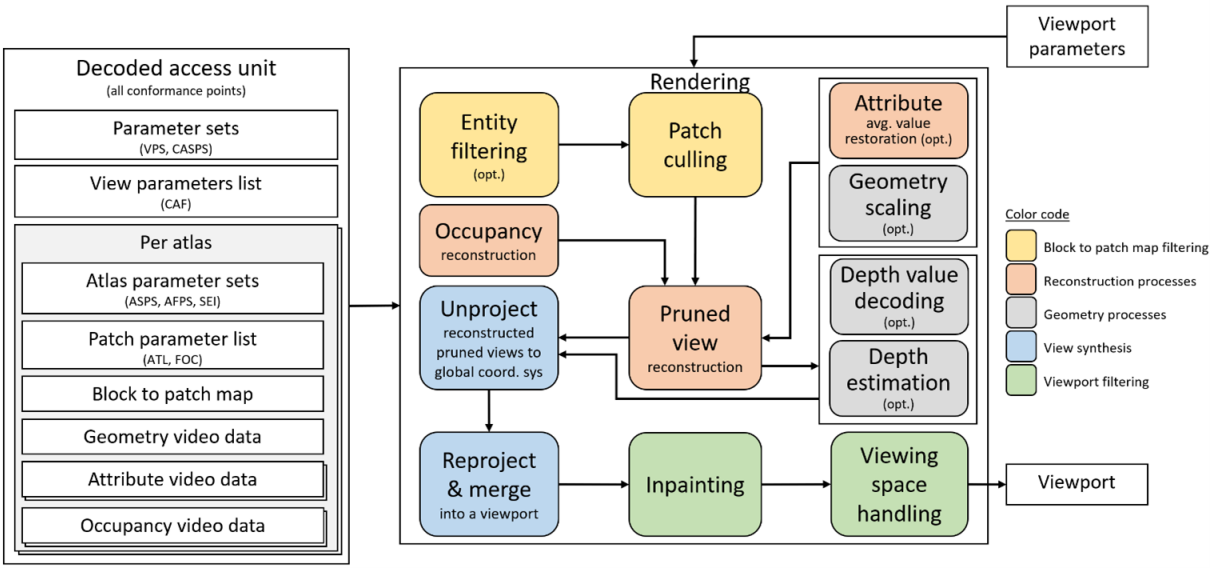

Figure 3.20: TMIV render pipeline

The rendering process in TMIV involves several stages, each playing a crucial role in reconstructing the viewport. The process begins with block to patch map filtering, which includes entity filtering and patch culling to reduce the data volume and enhance rendering efficiency. After that, the reconstruction stage commences, consisting of occupancy reconstruction, attribute average value restoration, and pruning view reconstruction. Next, the geometry process comes into play, encompassing geometry scaling, depth value processing, and depth estimation. Following this, the view synthesis phase involves unprojection, reprojection, and merging to generate the desired viewpoints. Finally, the viewport filtering step, which includes inpainting and view space handling, helps refine the rendered output. Throughout next subsections, we will delve into the critical operations and techniques used in rendering pro-

cess.

The output of the TMIV renderer is a viewport, which can be either a perspective view or an omnidirectional view. The rendered output includes both the texture attribute and geometry components. This viewport can be displayed using a variety of devices, such as a head-mounted display (HMD) or a regular 2D monitor screen equipped with a tracking system.

3.2.4.1 Entity Filtering

Entity filtering is an optional step in the rendering process, and it is performed only if the entity mode has been selected in the encoder side. The purpose of entity filtering is to reduce the amount of data that needs to be rendered by focusing on a specific subset of objects or entities in the scene. For example, the renderer may choose to render only the foreground objects, ignoring the background elements. To achieve this, the renderer selects the patches that belong to the target entities and includes them in the rendering process.

3.2.4.2 Patch Culling

In patch culling, the renderer discards patches that have no overlap with the requested target viewport. This process is performed to reduce the computational cost during the rendering phase. To determine whether a patch overlaps with the target viewport, the renderer reprojects the four corners of the patch to the target view using the geometry maps. By culling patches that are not visible in the target viewport, the renderer avoids unnecessary computations for those patches, leading to improved rendering performance and efficiency.

3.2.4.3 Occupancy Reconstruction

During this phase, the occupancy map is reconstructed, and there are three possible scenarios to consider. The first scenario occurs when the occupancy map is embedded within the geometry map. To extract the occupancy map, the renderer compares the geometry pixel values after performing the geometry upscaling process against a threshold value, which was previously signaled during the encoding phase as ”depthOccMapThreshold.” If the threshold value is smaller or equal to the geometry value at a given pixel, the occupancy value at that pixel is set to 1; otherwise, it is set to 0.

The second scenario arises when the occupancy video sub-bitstream is present. In this case, the renderer performs a thresholding process to reconstruct the occupancy map from

the decoded one.

The third scenario occurs when no occupancy map is explicitly signaled during the encoding phase. In this scenario, the occupancy map is built, and all its values are set to 1, indicating that the entire atlas is considered fully occupied.

3.2.4.4 Prune View Reconstruction

During this phase, the source views will be reconstructed by collecting all patches that belong to them from the different atlases. This process is the inverse of the patch packing operation that was performed during the encoding phase. The renderer retrieves the patches from the atlases and assembles them together to reconstruct the original source views. An example of this view reconstruction process is illustrated in Figure 3.21, where three views are reconstructed from the information stored in the atlases.

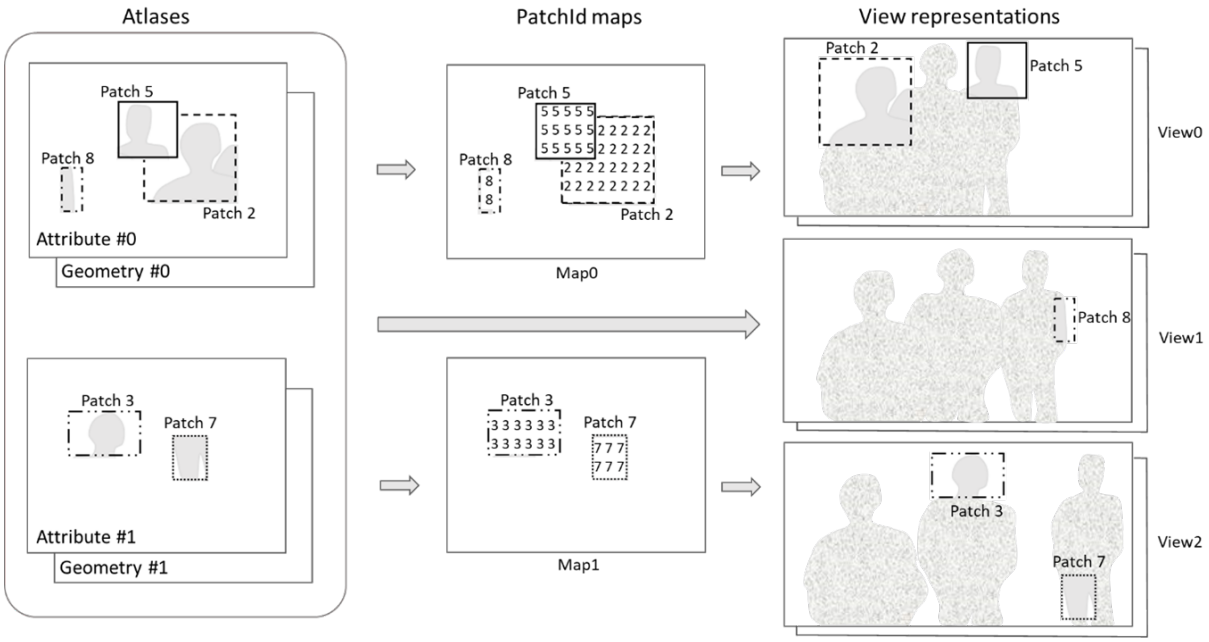

Figure 3.21: Prune view reconstruction

3.2.4.5 Depth Estimation

When TMIV operates in the geometry absent profile, only the texture atlases are encoded, and there are no geometry maps included in the decoded bitstream. Therefore, the depth maps need to be estimated on the decoder side to facilitate the rendering process. To achieve this, the decoder uses reconstructed pruned views along with their associated view parameters.

It's important to note that the TMIV software itself does not have an internal depth estimator. Instead, external software such as Immersive Video Depth Estimation (IVDE) or Depth

Estimation Reference Software (DERS) can be employed to estimate the depth maps. Once the depth maps are estimated, they are combined with the reconstructed pruned views and their respective view parameters [21].

With the estimated depth maps, reconstructed pruned views and associated view parameters, the TMIV renderer can proceed with the rest of the rendering operations, to generate the final viewport output.

3.2.4.6 View Synthesis

TMIV offers two methods for synthesizing views: the Reference View Synthesizer (RVS) and the View Weighting Synthesizer (VWS).

The Reference View Synthesizer (RVS) involves several steps in the view synthesis process. It begins with the reprojection phase, where the image is first unprojected into scene coordinates using the intrinsic camera parameters of the source view. Then, a change of reference frame is performed from the source view to the target view using the extrinsic camera parameters, which includes rotation and translation. Next, the scene coordinates are projected back into image coordinates using the intrinsic camera parameters of the target view. After reprojection, the renderer performs and finally blends the views as the last stage of the process.

The RVS synthesizer directly generates views from the atlases, eliminating the need for the view reconstruction phase. To render directly from the atlases, the renderer takes multiple texture attribute atlases and geometry atlases as inputs, along with block to patch maps per atlas, atlas parameters list, and camera parameters lists. Additionally, the target camera parameters are provided, which can represent either a perspective viewport or an omnidirectional view. The output of the renderer is a single view (viewport) [11] [22].

The View Weighting Synthesizer (VWS) performs two main operations: visibility and shading.

In the **visibility** operation, the VWS aims to generate a geometry map specifically for the target viewport. To achieve this, several steps are involved. First, warped geometry maps are generated for each input view. This is done by unprojecting the pixels from the source view and then reprojecting them onto the target view. Splat-based rasterization is used for this process instead of triangulation [23]. After generating the warped geometry maps for all input views, they are combined to produce a single geometry map known as the visibility map. This visibility map indicates which pixels are visible from the target viewport. During the generation of the visibility map, each view is given a weight using a weighting strategy.

The second operation in VWS is **shading**. This step involves calculating the color for the target view. Each pixel from the input views is blended into the target viewport using weights that consider its consistency with the visibility map and the weight assigned to the view it belongs to.

In the visibility and shading processes, a **weighting strategy** is employed to assess the weight of non-pruned pixels from each input view. Let's consider an example of how this is done:

For a non-pruned pixel 'P' from a view associated with node 'N' in the pruning graph, its initial weight W_P is determined based on its distance from the view being synthesized. This weight is denoted as W_N and is updated as follows:

If pixel 'P' reprojects into one of the pruned pixels belonging to child views, its weight is accumulated with the weight W_O of that child view, resulting in $W_P = W_P + W_O$. This process is then recursively repeated for the grandchildren.

If pixel 'p' does not reproject into any of its child views, the previous rule is extended recursively to the grandchildren as well.

If the pixel 'p' reprojects into one of its child views at an unpruned pixel, its weight is left unchanged, and no further inspection of the graph is performed towards the grandchildren.

By applying these rules recursively, the weights of non-pruned pixels are calculated based on their relationships within the pruning graph. The weighting strategy ensures that the contributions of each pixel from the input views are appropriately assessed and used in the visibility and shading operations. Figure 3.22 provides an illustration of the graph-based pruning process [11].

3.2.4.7 Inpainting

The inpainting process is essential for filling the holes in virtual views. For each empty pixel, the algorithm locates the two nearest neighbors on the left and right sides. The color of the empty pixel is then determined as a weighted average of these two neighboring pixels, with the weights being based on their distances from the empty pixel. However, in cases where there is a significant difference in the geometry values of the two neighbors, the texture attribute of the neighbor with geometry information is copied instead of using the weighted average.

When projecting Equi-Rectangular Projection (ERP) images to their respective views, horizontal inpainting can result in unnaturally lines, which are visually undesirable. To address this issue, an additional step is performed by changing the projection type to generate trans-

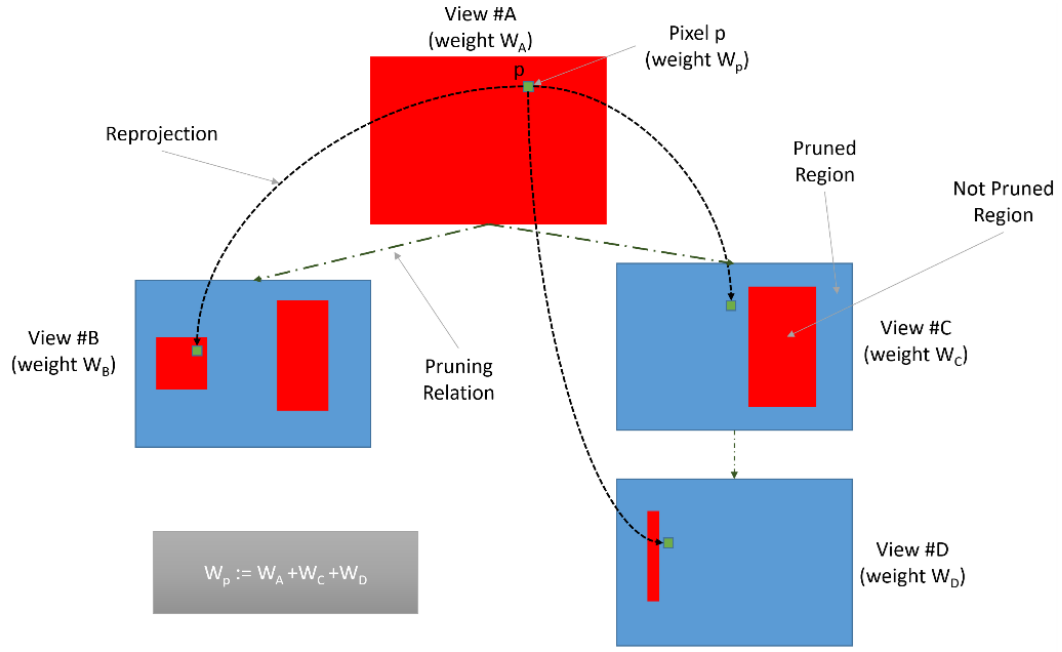

Figure 3.22: Graph-based pruning process

verse ERP images. This involves adjusting the length of all rows in an ERP image to match the circumference of the corresponding circle on a sphere, as depicted in Figure 3.23a. In a second step, all columns of the image are expanded to have the same length, as illustrated in Figure 3.23b, resulting in a transverse ERP image shown in Figure 3.23c [11].

The purpose of these inpainting techniques is to produce seamless and visually coherent virtual views by filling in missing information in a manner that preserves the overall quality and realism of the rendered content.

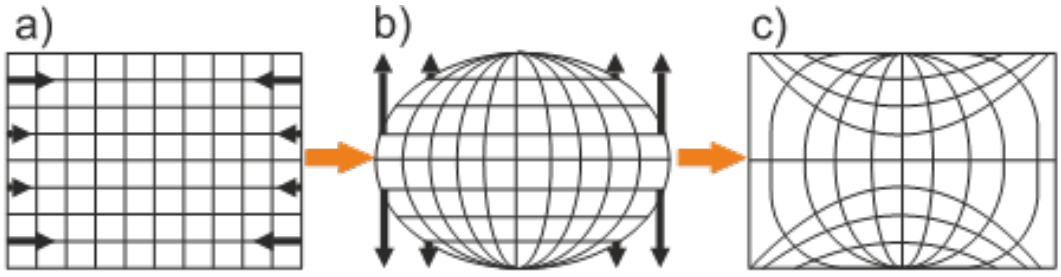

Figure 3.23: Equirectangular projection to transverse equirectangular projection

3.3 Common Test Conditions

MIV common test conditions (MIV-CTC) offer standardized test sequences and a structured pipeline for encoding and decoding processes, along with a methodology to evaluate coding efficiency. MIV-CTC provides a clear framework for conducting coding experiments in a controlled environment, enabling easy and reliable comparison of the experimental results [20] [24].

3.3.1 Test Sequences

CTC offers a diverse collection of 21 test sequences, encompassing both computer-generated and natural scenes, along with estimated depth maps. The sequences are captured using various camera types, including both perspective cameras and omnidirectional (ERP) cameras. The number of views in the sequences ranges from 9 to 25, and the resolutions vary from full HD to 4K. Additionally, there is a wide variety of camera arrangements, such as linear camera setups and array camera configurations.

Table 3.1 provides the list of sequences which are organized into two main categories: natural and computer-generated content (CG). Within these categories, the sequences are further classified into different classes, each representing a specific camera type:

- Class A: Omnidirectional scenes captured by spherical cameras (ERP representation).
- Class B: Omnidirectional scenes captured by semi-spherical cameras (ERP representation).
- Class C: Semi-omnidirectional scenes captured by semi-spherical cameras (ERP representation).
- Classes D and J: Scenes captured by camera arrays (perspective content).
- Class E: Scenes captured by linear multicamera systems (perspective content).
- Classes L and W: Scenes captured by converging cameras (perspective content).

3.3.2 Anchor Definition

CTC provides two anchor conditions for encoding multi-view sequences. The first anchor is the MIV anchor, which handles both attribute and geometry source views. In this anchor, some source views are fully packed, while only patches from other views are included. The second anchor is the MIV decoder-side depth estimating anchor, where we work with only the attribute of source views without any geometry information. Here, a subset of complete source views is packed without encoding geometry, and depth estimation is performed on the decoder side.

In addition to the anchors, there is a non-anchor reference condition called ”best reference.” In this condition, the content is directly rendered from all source views without any coding. This approach uses a renderer with the view weighting synthesizer (VWS) to achieve the rendering process without compression.

Table 3.1: List of sequences

Computer generated content		Natural content	
Class A:		Class D:	
A01	ClassroomVideo	D01	Painter
Class B:		D02	Breakfast
B01	Museum	D03	Barn
B02	Chess	Class E:	
B03	Guitarist	E01	Frog
Class C:		E02	Carpark
C01	Hijack	E03	Street
C02	Cyberpunk	Class L:	
Class J:		L01	Fencing
J01	Kitchen	L02	CBABasketball
J02	Cadillac	L03	MartialArts
J03	Mirror		
J04	Fan		
Class W:			
W01	Group		
W02	Dancing		

Table 3.2: List of texture QPs for the MIV anchor

Sequence		RP1	RP2	RP3	RP4
A01	ClassroomVideo	26	30	38	51
B01	Museum	29	40	47	51
B02	Chess	17	27	35	45
J01	Kitchen	17	26	33	41
W02	Dancing	19	23	28	40
D01	Painter	21	32	43	51
E01	Frog	28	34	40	46

3.3.2.1 Coding for the MIV anchor

CTC provides four rate points (RP) or quantization parameters (QP) for each sequence. The subset of texture QPs is shown in table 3.2. For every texture QP, there corresponds a single geometry QP. The geometry QP (q') can be calculated from attribute QP (q) using the following equation for the entire sequences, [.] indicates rounding to the nearest integer operation. Apart from the four rate points mentioned earlier, there is an additional rate point called "RP0" where no video compression is applied.

$$q' = \max\left(1, [-14.2 + .8q]\right) \tag{3.2}$$

Table 3.3: List of texture QPs for the MIV DSDE anchor.

Sequence		RP1	RP2	RP3	RP4
A01	ClassroomVideo	29	33	41	50
B01	Museum	39	47	49	51
B02	Chess	23	29	35	42
J01	Kitchen	26	32	38	43
W02	Dancing	28	33	38	42
D01	Painter	23	30	36	41
E01	Frog	29	32	37	44

To encode the MIV anchor, the following steps need to be followed:

1. Use the TMIV encoder to generate the TMIV bitstream along with the attribute and geometry atlases.

2. Encode the attribute and geometry atlases using an agnostic video coding method, such as VVenC (Fraunhofer Versatile Video Encoder), which is a fast and efficient H.266/VVC encoder that is used here.

3. Multiplex the TMIV bitstream with the video sub-bitstreams generated in the previous step.

4. Decode and render the bitstream using the TMIV decoder.

3.3.2.2 Coding for Decoder-side depth estimating anchor

The MIV DSDE anchor operates in the geometry absence profile, which means it deals with attribute source views only, without any geometry source views. In this anchor, a separate table for quantization parameters (QPs) is provided for textures, but there are no QPs for geometry since no geometry maps are present.

Similar to the MIV anchor, there is an additional rate point called ”RP0” for the MIV DSDE anchor, where no video compression is applied.

Table 3.3 provides a list of texture QPs for the MIV DSDE anchor for some sequences.

To encode the DSDE anchor, the following steps need to be followed:

1. Use the TMIV encoder to generate the TMIV bitstream along with the attribute atlases.

2. Encode the attribute atlases using an agnostic video coding method, such as VVenC encoder which is used here.

3. Multiplex the TMIV bitstream with the video sub-bitstreams generated in the previous step.

4. Decode the bitstream using the TMIV decoder, no rendering in this step.

5. Estimate the depth maps using any depth estimators, here CTC use Immersive Video Depth Estimation (IVDE) software.

6. Use TMIV renderer to render the results from previous steps.

3.4 Conclusion

The MIV standard represents a cutting-edge approach for encoding volumetric data captured by multiple cameras, each providing a unique viewpoint. By leveraging years of video coding advancements, MIV optimally utilizes the benefits of established video coding techniques within this new standard. In this chapter, we have delved into the intricacies of the MIV standard, we explored the Test Model (TMIV) proposed by the MPEG group, which offers a broader perspective on the encoding and decoding pipeline. From the encoder to the bitstream, decoder, and renderer, the entire process has been thoroughly explained.

Additionally, we discussed the significance of Common Test Conditions (CTC), which introduces standardized test sequences with diverse characteristics and a structured pipeline for encoding and decoding, along with a comprehensive evaluation methodology for coding efficiency. Furthermore, we explored different types of anchors, such as the MIV anchor and DSDE anchor, and provided a brief overview of how they function.

In the upcoming chapter, we will delve into practical implementations of the concepts discussed in this chapter using the TMIV software, as proposed by the MPEG group in the MIV standard.

Chapter 4

Implementation

4.1 Introduction

In the last chapter, we took a deep dive into the MIV standard, thoroughly discussing the proposed test model (TMIV). This model provided a comprehensive framework that covered both encoding and decoding procedures. We covered various aspects, including atlas creation, understanding the MIV bitstream, decoding strategies, rendering techniques, and the significance of following the Common Test Conditions (CTC) for experimental consistency.

However, everything covered in the previous chapter was more on the theoretical side. In this chapter, we are shifting gears to practical implementation. We will walk through the concepts discussed earlier but in a hands-on manner. To achieve this, we will use TMIV software which had been proposed by the MIV standard. We will carry out TMIV encoding, decoding, and rendering processes. Additionally, we will employ VVenC for video coding.

This chapter will be divided into three main sections. First, we will delve into the sequences that we will be working with throughout the process. Next, we will explore the implementation of the MIV anchor in detail. Finally, we'll explore the Best Reference method, a technique that directly renders the viewport from the input views without involving any compression.

4.2 Sequences

In the implementation part, five sequences were used with different projection types, the sequences are Frog, Chess, Painter, Carpark, and ClassroomVideo. However, the detailed explanation will focus on two of these sequences which are the Frog sequence and the Chess sequence. The "Frog" sequence used rectilinear projection, which maps a surface of a sphere

Table 4.1: Characteristics of the Frog sequence.

Category – Short name	E01
Input contribution	WG 11 M43748, WG 11 M44914 and WG 11 M47445
Length & frame rate	300 frames (30 fps)
Number of source views	13 (13x1)
Source view resolution	1920x1080
Texture format	YUV 4:2:0 10 bits
Depth format	YUV 4:2:0 16 bits
Depth range	[0.3 m, 1.62 m], normalized disparity
View FoV & mapping	63.65° × 38.47° Rectilinear
Lens	2.16 mm
Camera spacing	3.675 cm

to a flat image, rectilinear projection here maps the distorted image captured by the lens onto a flat plane in such a way that the distortion is removed. The “Chess” sequence used equirectangular projection, in which the images are mapped onto a rectangular grid, where each pixel’s position corresponds to a specific latitude and longitude on the spherical surface.

The ”Frog” sequence consists of 13 source views, including both texture and geometry videos. The cameras capturing these views are arranged linearly from left to right. It is denoted by the short name ”E01” and has a resolution of 1920x1080 pixels. The texture videos are encoded in YUV 4:2:0 10-bit format, while the geometry videos are encoded in YUV 4:2:0 16-bit format. This sequence contains 17 frames with a frame rate of 30 frames per second. For more detailed characteristics of the ”Frog” sequence, please refer to Table 4.1. An example of texture and geometry images taken from source view numbers 1, 4, and 8, denoted as ”V1” ”V4” and ”V8” is shown in Figure 4.1.

The ”Chess” sequence consists of 10 source views, which include both texture and geometry videos. The cameras capturing these views are arranged spherically, as depicted in Figure 4.2. It is denoted by the short name ”B02” and has a resolution of 2048x12048 pixels. The texture videos are encoded in YUV 4:2:0 10-bit format, while the geometry videos are encoded in YUV 4:2:0 16-bit format. This sequence comprises 17 frames with a frame rate of 30 frames per second. For more detailed characteristics of the ”Chess” sequence, please refer to Table 4.2. An example of texture and geometry images taken from source view numbers 1, 4, and 8 is shown in Figure 4.3.

The sequences should be saved under the ”content” folder inside the ”workspace” project folder, and they are stored in YUV format. The names of the views follow a specific spatial order as follows: ”v{i}_{t}_{w}x{h}_yuv{f}p{b}le.yuv”, where:

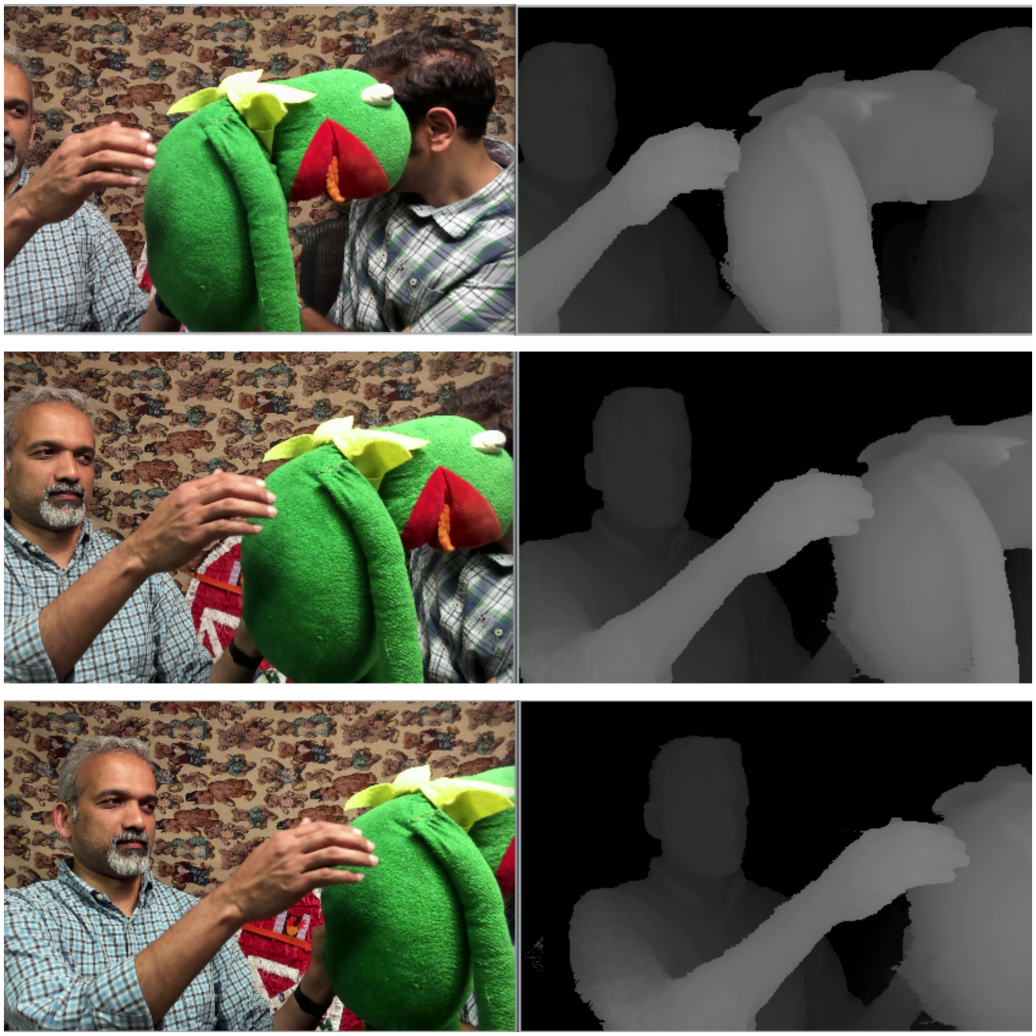

Figure 4.1: Frog sequence, image taken by ”v1”, ”v4” and ”v8” source views, texture images on the left, geometry maps in the right

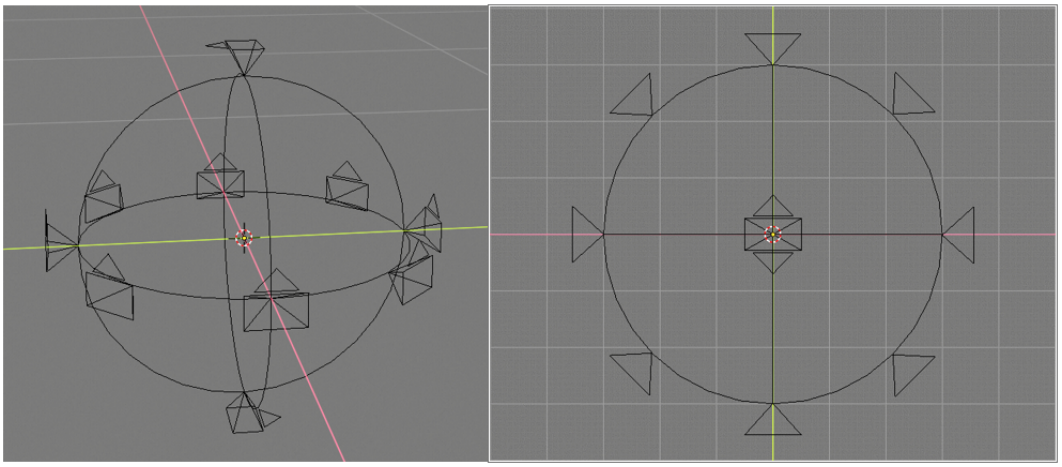

Figure 4.2: Chess sequence camera’s arrangement

Table 4.2: Characteristics of the Chess sequence.

Category – Short name	B02
Input contributions	WG 11 M50787
Length & frame rate	300 frames (30 fps)
Number of source views	10
Texture format	YUV 4:2:0 10-bits
Depth format	YUV 4:2:0 16-bits
Depth range	[0.1 m, 500 m], normalized disparity
Source view resolution	2048 × 2048
View FoV & mapping	180° × 180° ERP
Global FoV	360° × 180°

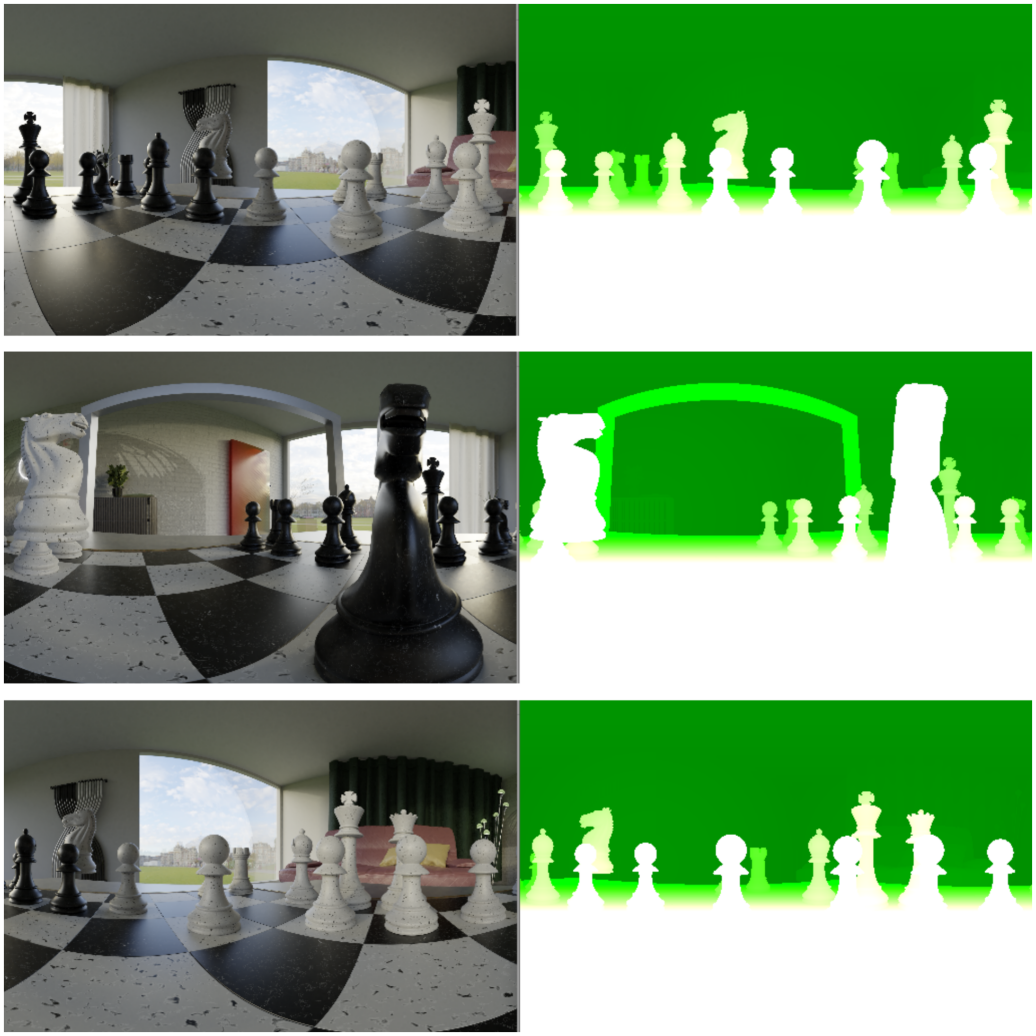

Figure 4.3: Chess sequence, image taken by ”v1”, ”v4” and ”v8” source views, texture images in the left, geometry maps on the right 44

- i: is an integer number to identify the camera's number.
- t: specifies if the video is a texture or depth map.
- w: is the width of the luma channel of the video.
- h: is the height of the luma channel of the video.
- f: denotes the YUV subsampling format.
- b: indicates the number of bits per channel.

For example, the names of the sequences could be "v1_depth_2048x2048_yuv420p16le" and "v1_texture_2048x2048_yuv420p10le", representing depth and texture videos respectively captured by camera "v1" with a resolution of 2048x2048 pixels, encoded in YUV 4:2:0 subsampling format with 16 bits per channel for depth and 10 bits per channel for texture.

4.3 MIV Anchor

The MIV Anchor, also known as the MIV Main Anchor, is responsible for dealing with basic views and patches. This mode requires the presence of texture source views and geometry maps. To implement the MIV Anchor for a sequence, the following steps need to be followed using the TMIV software proposed by the MPEG group:

1. Encode the sequence using the TMIV encoder to generate the MIV bitstream and texture and geometry atlases.
2. Encode the attribute and geometry atlases using agnostic video coding. VVenC (Fraunhofer Versatile Video Encoder), which is a fast and efficient H.266/VVC encoder used in this process.
3. Decode the video bitstreams to recover the attribute and geometry atlases.
4. Use the TMIV decoder to decode and render the desired viewport from the atlases.

Each of these steps will be explained in detail in the following subsections to understand how the MIV Anchor works.

4.3.1 TMIV Encoder

The aim of this stage is to generate atlases from the given input sequence. These atlases are composed of basic views that can be directly included in the atlases. Additionally, there are additional views that go through a pruning process utilizing a pruning graph. This process generates patches which are subsequently integrated into the atlases after the basic views have been packed. To achieve this, the TmivEncoder.exe tool included in the TMIV software is utilized. The Command Prompt (cmd) is used to execute the encoding process for the "Frog" sequence using the following command:

```
D:\Workspace\tmiv_install\bin\TmivEncoder -n 17 -s E01 -f 0
-c D:\Workspace\tmiv\config\ctc\miv_main_anchor\A_1_TMIV_encode.json
-p configDirectory D:\Workspace\tmiv\config
-p inputDirectory D:\Workspace\Content
-p outputDirectory D:\Workspace\Experiment
```

The parameters used in the command are explained as follows:

- -n: Specifies the number of frames in the input sequence, which is 17 frames for this sequence.
- -s: Indicates the content ID, "E01" is assigned for the "Frog" sequence.
- -f: Specifies the starting frame number, which in this case is frame 0.
- -c: Provides the path to the configuration file, where the file named "A_1_TMIV_encode.json" is used for configuration settings.
- -p: Used to specify different folders in the process. For instance, it is used to provide the path to the configuration folder, the input directory (where the sequence data is stored, i.e., "Content" folder), and the output directory where the results will be saved (i.e., "Experiment" folder).

The outcome of this procedure will yield four atlases in YUV format, comprising atlases for textures and two for depth maps, along with the TMIV bitstream. The results of this phase are illustrated in Figure 4.4, which displays the attribute and geometry atlases.

As evident from Figure 4.4, four views have been packed into atlases as basic views, whereas the remaining views have been pruned and subsequently packed into atlases as additional views.

Figure 4.4: Four atlases generated as a result of TMIV encoder (Frog Sequence)

The identical process was applied to the ”Chess” sequence (equirectangular projection) with a change in the content ID to ”B02”. The results of the atlases generated through TMIV encoding for this sequence are depicted in Figure 4.5.

Furthermore, the TMIV encoder generates a TMIV bitstream as an output. This bitstream contains essential information regarding the arrangement of views and patches within the atlases. It is commonly utilized in the decoder phase to reconstruct the source views or for rendering purposes to directly render the desired views from the atlases.

4.3.2 VVenC Encoder

The purpose of this step is to generate video sub-bitstreams from the video atlases generated in the previous step. These video atlases are encoded using agnostic video coding. here VVenC encoder had been used, which is utilized in the Common Test Conditions (CTC). The process involves using the vvencFFapp.exe tool, provided by the TMIV software, in the Command Prompt (cmd) to encode the Frog sequence. The following command is an example used to encode texture atlases:

```
D:\Workspace/tmiv_install/bin/vvencFFapp
-c D:/Workspace/tmiv/config/ctc/miv_main_anchor/A_2_VVenC_encode_tex.cfg
```

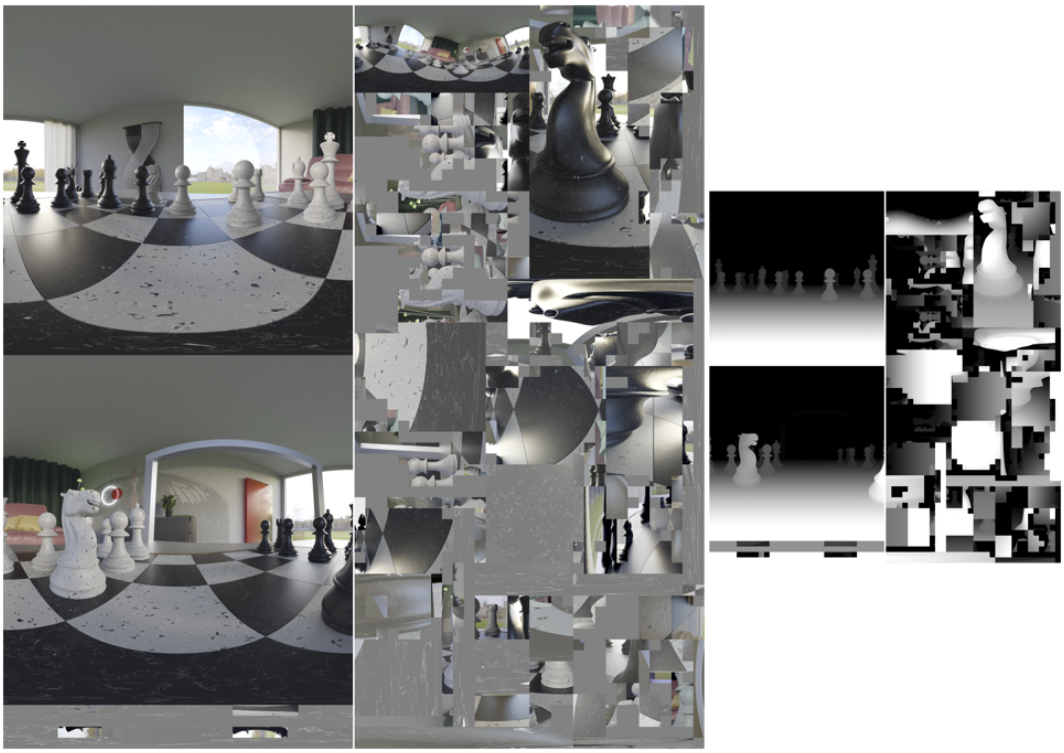

Figure 4.5: Four atlases generated as a result of TMIV encoder (Chess Sequence)

```
-i D:\Workspace\Experiment\A17\E01\RP0/TMIV_A17_E01_RP0_tex_c00_1920x4608_yuv420p10le.yuv
-b D:/Workspace/Experiment/A17/E01/QP2/TMIV_A17_E01_QP2_tex_c00.bit
-s 1920x4608 -q 34 -f 17 -fr 30
```

The parameters used in the command are explained as follows:

- -c: Specifies the path to the VVenC configuration file, for example to encode texture we used the file named "`A_2_VVenC_encode_tex.cfg`", which contains configuration settings for the VVenC encoder for textures. When encoding geometry, we used "`A_2_VVenC_encode_geo.cfg`".
- -i: Specifies the path to the input atlas file, which needs to be encoded.
- -b: Specifies the path to the output video sub-bitstream file with a name which will be a result from the encoding process.
- -s: Specifies the resolution (width*height) of the provided input atlas.
- -q: Specifies the desired quantization parameter (QP) for encoding. This parameter is determined based on each sequence (Table 3.2). The QP value for textures is provided directly, while the QP value for geometry needs to be calculated using formula 3.2. For the Frog sequence QP2 had been used, which QP value of 34 is used for textures, and 13 for geometry.

- -f: Specifies the number of input frames, which in this case is frame 17.
- -fr: Specifies the frame rate of the atlases, which is 30 frames per second.

Executing this command encodes a single video sub-bitstream, and this process needs to be repeated for each atlas, resulting in a total of four encoded sub-bitstreams. An example of the command used for encoding the geometry atlas is shown below:

```
D:\Workspace/tmiv_install/bin/vvencFFapp
-c D:/Workspace/tmiv/config/ctc/miv_main_anchor/A_2_VVenC_encode_geo.cfg
-i D:\Workspace\Experiment\A17\E01\RP0/TMIV_A17_E01_RP0_geo_c01_960x2304_yuv420p10le.yuv
-b D:/Workspace/Experiment/A17/E01/QP2/TMIV_A17_E01_QP2_geo_c01.bit
-s 960x2304 -q 34 -f 17 -fr 30
```

4.3.3 VVdeC Decoder

After encoding our video sub-bitstreams using VVenC, the next step involves performing the inverse operation in the decoder to recover our atlases for later phases such as TMIV decoding and rendering. To achieve this, we use the vvdecFFapp.exe executable provided by the TMIV software. The following command is used in the Command Prompt (cmd) to decode the video sub-bitstreams:

```
D:/Workspace/tmiv_install/bin/vvdecapp
-b D:/Workspace/Experiment/A17/E01/QP2/TMIV_A17_E01_QP2_tex_c00.bit
-o D:\Workspace\Experiment\A17\E01\QP2/TMIV_A17_E01_QP2_tex_c00_1920x4608_yuv420p10le.yuv
```

The parameters used in the command are explained as follows:

-b: Specifies the path of the input video sub-bitstream file that needs to be decoded. -o: Specifies the path for the output video atlas, which will be the result of the decoding process.

Executing this command decodes a single video sub-bitstream, and this process needs to be repeated for each atlas, resulting in a total of four video atlases.

4.3.4 TMIV Decoder

The objective of this phase is to restore the original input views from the atlases, but the outcomes will differ based on the specific anchor in use.

When employing the MIV anchor, the TMIV decoding process is combined with the rendering process. This yields the desired target view as well as an additional view known as the ”pose trace view.” The term ”pose trace view” generally refers to a visual representation that illustrates the trajectory or path followed by a user’s or camera’s pose (position and orientation) over a specific time frame. This view is used to simulate movements in a 3D space

and is often employed for subjective quality assessment. In the TMIV software, each sequence is accompanied by three pose trace views labeled as ”p01,” ”p02,” and ”p03,” each simulating distinct movements by users or cameras.

On the other hand, when TMIV decoding is utilized in conjunction with the decoder side depth estimation anchor, the TMIV decoder recovers the original source views from the atlases. Subsequently, the TMIV renderer can be used to generate both the target view and the pose trace view.

To perform the TMIV decoding process, the TMIV bitstream generated during the TMIV encoding process is utilized, along with the atlases generated in the previous phase. The TMIV decoder.exe provided by the TMIV software is employed, and the following command is executed in the Command Prompt (cmd) to render the V11 target view and pose trace view using the ”p02” pose trace:

```
D:/Workspace/tmiv_install/bin/TmivDecoder -n 17 -N 51 -s E01 -r QP2 -v v11 -P p02
-c D:/Workspace/tmiv/config/ctc/miv_main_anchor/A_4_TMIV_decode.json
-p configDirectory D:/Workspace/tmiv/config
-p inputDirectory D:\Workspace\Experiment
-p outputDirectory D:\Workspace\Experiment
```

The parameters used in the command are explained as follows:

- -n: Specifies the number of frames in the input sequence, which is 17 frames for this sequence.
- -N: Specifies the number of desired output frames. For the target view, the number of output frames will be the same as the number of input frames. However, in the pose trace view, the number of output frames can be looped, and it can exceed the number of input frames. In this case, the number of frames for the target view will be 17, and the number of frames for the pose trace view will be 51.
- -s: Indicates the content ID. ”E01” is assigned for the ”Frog” sequence.
- -r: Indicates the test ID. It is used to label various video encodings carried out under different configurations, such as QP1, QP2, and so forth, or R0 for lossless compression. In this case, QP2 is used.
- -v: Indicates the view name of the target view. Here, we specify V11 as our target view.
- -P: Indicates the pose trace number that we want to use. In this case, we use p02 as our pose trace.

- -c: Provides the path to the configuration file. The file named `A_4_TMIV_decode.json` is used for configuration settings.
- -p: Used to specify different folders in the process. For instance, it's used to provide the path to the configuration folder, the input directory (where the atlases are stored, i.e., "Experiment" folder), and the output directory where the results will be saved (i.e., "Experiment" folder).

The outcome of the TMIV decoder/renderer applied to the frog sequence includes the V11 target view and the pose trace view using the "p02" pose trace. The rendered target view is depicted in Figure 4.6, while Figure 4.7 illustrates the rendered pose trace view.

Figure 4.6: Rendered target view "V11" using MIV anchor (Frog sequence)

The same procedure was conducted for the chess sequence, where the V5 target view was utilized, accompanied by the "p01" pose trace. The outcome of the chess target view is presented in Figure 4.8, while the result of the chess pose trace is displayed in Figure 4.9.

4.4 Best Reference

In addition to the MIV anchor and decoder side depth estimation anchor, the Common Test Conditions (CTC) also include a non-anchor reference method known as the "best reference." This approach involves direct rendering from all input source views without the need for coding, essentially utilizing the original input sequence. In the best reference scenario, the TMIV renderer is employed in conjunction with the view weight synthesizer (VWS).

Figure 4.7: Rendered view using pose trace P02 using MIV anchor (Frog sequence)

Figure 4.8: Rendered target view ”V5” using MIV anchor (Chess sequence)

Figure 4.9: Rendered view using pose trace P01 using MIV anchor (Chess sequence)

To carry out the best reference rendering, the TmivRenderer.exe tool provided by the TMIV software is utilized. The following command is executed in the Command Prompt (cmd) to render the V11 target view and the "p02" pose trace view using the Frog sequence:

```
D:/Workspace/tmiv_install/bin/TmivRenderer -n 17 -N 51 -s E01 -r R0 -v v11 -P p02 -f 0
-c D:/Workspace/tmiv/config/ctc/best_reference/R_1_TMIV_render.json
-p configDirectory D:/Workspace/tmiv/config -p inputDirectory D:/Workspace/Content
-p outputDirectory D:/Workspace/Experiment
```

The TMIV renderer shares similar input with the TMIV encoder and produces output that aligns with the TMIV decoder.

The command's parameters are elucidated as follows:

- -n: Specifies the number of frames in the input sequence, which for this case is 17 frames.
- -N: Specifies the number of desired output frames. For the target view, the output frame count matches the input frame count. In the pose trace view, the number of output frames can loop and exceed the input frame count. Here, the target view has 17 frames, and the pose trace view has 51 frames.
- -s: Represents the content ID, with "E01" assigned to the "Frog" sequence.
- -r: Denotes the test ID, used to label various video encodings under different configurations (e.g., QP1, QP2, R0 for lossless). Here, R0 is utilized.
- -v: Specifies the target view's name. In this instance, V11 is designated.
- -P: Identifies the pose trace number to use. In this case, p02 is employed.

- -f: Specifies the starting frame number, which is 0 in this scenario.
- -c: Provides the path to the configuration file. The file named `R_1_TMIV_render.json` is used for configuration settings.
- -p: Specifies various folders in the process, including the configuration folder, the input directory (holding the source views, i.e., ”Content” folder), and the output directory (where the results are stored, i.e., ”Experiment” folder).

Utilizing the TMIV renderer for the best reference approach in the frog sequence yields the V11 target view and the pose trace view using the ”p02” pose trace. These rendered views are visually represented in Figures 4.10 and 4.11. Similarly, the best reference technique was applied to the chess sequence, resulting in Figure 4.12 showcasing the target view V5 as the outcome of this process, and Figure 4.13 displaying the pose trace view.

Figure 4.10: Rendered target view ”V11” using Best Reference (Frog sequence)

4.5 Conclusion

The MIV standard has proven to be a successful approach for compressing volumetric data represented in the MVD format. The MIV standard introduced a test model to comprehensively explain the intricate processes of encoding and decoding. Additionally, to translate these concepts into practice, the TMIV software was developed to provide a practical implementation of the MIV standard’s principles and the test model.

Figure 4.11: Rendered pose trace view ”P02” using Best Reference (Frog sequence)

Figure 4.12: Rendered target view ”V5” using Best Reference (Chess sequence)

Figure 4.13: Rendered pose trace view ”P01” using Best Reference (Chess sequence)

Throughout this chapter, we delved into the details of the MIV anchor and the Best Reference mode. In our practical implementation, we utilized various executable applications provided by the TMIV software. For instance, the TmivEncoder application was employed to encode input views, creating atlases and the MIV bitstream. We used vvencFFapp to encode video atlases through VVdeC, vvdecapp to recover atlases, and finally, TmivDecoder to decode and render the viewport. Furthermore, we explored the Best Reference mode, which involves rendering directly from input views and can serve as a reference model for evaluation. Here, we utilized TmivRenderer for this purpose.

In the upcoming chapter, we will shift our focus to objective and subjective evaluation methodologies. We will explore how these evaluation techniques can be leveraged to assess the results generated in this chapter’s practical implementations.

Chapter 5

Quality Assessment

5.1 Introduction

In the realm of multimedia, especially with videos, transmitting large volumes of data to end users is a challenge due to bandwidth limitations. To address this, extensive research, including efforts by MPEG and other groups, has been dedicated to data reduction techniques. Nowadays, immersive media and volumetric data which involve very huge amounts of data are key players, enabling users to experience a sense of immersion with 6 degrees of freedom (6DoF) through augmented reality (AR) and virtual reality (VR) applications. The increasing amount of data shows the need to utilize the bitstream in an effective way.

Efficient utilization of bitstreams involves compressing data at the transmitter side and decompressing it at the receiver side. However, a major challenge lies in minimizing the potential loss in quality during this process. This makes quality assessment crucial. During compression and decompression, considering viewers' experiences becomes essential, and an understanding of the human visual system (HVS) is pivotal to achieving high-quality experiences (QoE). Quality of experience (QoE) is defined "the degree of delight or annoyance of the user of an application or service", both in a general context and particularly in immersive media technologies '[25]'[26] [27].

Quality assessment is crucial for achieving the highest level of user experience quality. In the context of volumetric data, the compression and decompression stages can potentially impact the overall quality. As a result, it's essential to evaluate the quality to ensure optimal outcomes. Two main types of evaluation methods are employed. The first is a subjective evaluation, which involves direct user engagement and is considered the most reliable approach. The second is objective evaluation, which employs mathematical calculations to assess quality.

In this chapter, we will delve into the definitions of subjective and objective evaluation, compare their characteristics, and highlight well-known methods utilized in both approaches. Additionally, we will provide a deeper insight into our quality assessment process especially objective evaluation, which was employed to evaluate the results generated using TMIV software from the previous chapter.

5.2 Subjective Evaluation

Subjective evaluation serves as a valuable method for assessing quality by gathering participants' opinions on specific experiments. A commonly employed approach in subjective evaluation is the mean opinion score (MOS), which can be defined as "The likely level of satisfaction of a service or product as appreciated by an average user"[28]. Participants are invited to rate the presented content using various methodologies, including single stimulus (SS), double stimulus (DS), and multiple stimulus setups [29].

In the single stimulus (SS) method, participants are exposed to one visual at a time, a methodology utilized in test methods like Absolute Category Rating (ACR) [30], Absolute Category Rating with Hidden Reference (ACR-HR) [30], and Single Stimulus Continuous Quality Evaluation (SSCQE) [31]. Alternatively, the double stimulus (DS) method involves simultaneous presentation of two visuals, with variations like Degradation Category Rating (DCR) [30], Double Stimulus Continuous Quality Evaluation (DSCQE) [31], and Double Stimulus Impairment Scale (DSIS)'[32] being employed [29].

In certain cases, multiple stimuli are presented concurrently, as seen in methods like Subjective Assessment Methodology for Video Quality (SAMVIQ) [33].

Upon gathering participant data, it is important to conduct data refinement processes, such as eliminating outliers and reducing noise. Subsequent statistical analyses, such as calculating means and standard deviations, provide a clearer insight into the collected data [29].

5.2.1 Subjective Evaluation for TMIV

As a result of rendering in the TMIV software, two types of views are generated: the target view, which is derived from input views or we can say regenerating a specific input view and used for objective evaluation, and the pose trace view, which is based on pose traces provided in the TMIV software and used for subjective evaluation.

Rendering the generated viewport and showing it using a head-mounted display allows

users to experience content with six degrees of freedom (6DoF), enabling them to view the content from different angles. However, evaluating quality using this method can be challenging due to the variability in user experiences. To address this, pose trace views are generated to simulate what users might see with limited motion provided by pose traces. These pose trace views can be displayed on a 2D screen, allowing for controlled view direction and facilitating subjective evaluation.

Pose traces define the position and orientation of the viewport for each frame to be synthesized. Each sequence is accompanied by three pose traces (Xp01, Xp02, and Xp03, with X denoting the sequence ID) provided as CSV files within the TMIV software's configuration folder. Each row in the file corresponds to position and orientation values for a specific frame. The file contains six columns: X, Y, Z (for position), and Yaw, Pitch, Roll (for orientation). The resulting pose trace view has a resolution of 1920x1080 pixels, a 90-degree field of view, and a 10-bit YUV format. Each file typically contains around 300 rows, representing values for frames. An illustrative example of a subset of these values is shown in Figure 5.1.

X	Y	Z	Yaw	Pitch	Roll
0.025176	0.001967	-0.00526	2.91505	-1.355	-2.91804
0.024877	-6.08E-05	-0.00469	2.894625	-1.3518	-2.89768
0.024579	-0.00206	-0.00411	2.8742	-1.34859	-2.87732
0.024249	-0.00417	-0.0035	2.856295	-1.34493	-2.85897
0.023855	-0.00649	-0.00283	2.83839	-1.34127	-2.84061
0.023386	-0.00905	-0.0021	2.82646	-1.34031	-2.82847
0.022833	-0.01185	-0.00131	2.81453	-1.33934	-2.81633
0.02219	-0.01483	-0.00049	2.800875	-1.33756	-2.80418
0.021453	-0.01791	0.000353	2.78722	-1.33577	-2.79202
0.02059	-0.02111	0.001214	2.77669	-1.33491	-2.78468
0.019569	-0.02448	0.002101	2.76616	-1.33405	-2.77733
0.018398	-0.02799	0.003014	2.75734	-1.33555	-2.77352
0.017085	-0.03164	0.003954	2.74852	-1.33704	-2.7697
0.015671	-0.03541	0.004913	2.74599	-1.34085	-2.77204
0.014198	-0.03924	0.005887	2.74346	-1.34466	-2.77438
0.01268	-0.04317	0.006884	2.7412	-1.34812	-2.77748
0.011135	-0.0472	0.007914	2.73894	-1.35157	-2.78057
0.009631	-0.0511	0.008919	2.736815	-1.35634	-2.78273
0.008236	-0.05462	0.009843	2.73469	-1.3611	-2.78489
0.006887	-0.05781	0.010702	2.729815	-1.36531	-2.78327

Figure 5.1: A subset of pose trace values (p02) for Frog sequence

While subjective evaluation through pose trace views is effective, but still this approach is not easy to do, expensive and it's time-consuming, because of that objective metrics come to play.

5.3 Objective Metrics Evaluation

Objective metrics aim to mimic subjective mean opinion scores through mathematical calculations. Various methods have been developed to assess quality. For example, peak signal-to-noise ratio (PSNR) and mean square error (MSE) calculate quality based on pixel values. Structural similarity index measure (SSIM) [34], multiscale structural similarity index measure (MS-SSIM) [35], and feature similarity index (FSIM) [36] assess structural aspects by incorporating the human visual system (HVS). Other methods, such as visual information fidelity (VIF) [37] and information fidelity criterion (IFC) [38], evaluate quality based on statistical characteristics [29].

Some metrics combine multiple features for quality estimation like video multi-assessment fusion (VMAF) [39] and video quality metric (VQM) [40]. Objective metrics can be categorized based on their approach. Full reference metrics, such as PSNR, SSIM, MS-SSIM, FSIM, VQM, and VMAF, compare two images to compute quality. No-reference metrics, including IFC and VIF, employ image statistics for quality calculation. The accuracy of quality estimation varies among metrics, but these methods provide a mathematical estimation of how users perceive results, offering a more manageable alternative to subjective evaluation.

It's worth noting that the complexity of these metrics can differ from one method to another, offering various levels of accuracy and practicality in quality assessment.

5.3.1 Peak Signal-to-Noise Ratio (PSNR)

Peak Signal-to-Noise Ratio (PSNR) is a widely used quality metric in image and video processing. It quantifies the difference between a reference image (original) and a distorted image (compressed or processed) in terms of signal noise. PSNR is calculated by comparing the maximum possible signal power to the power of the noise present in the image. It provides a numerical value that reflects the level of distortion or degradation in the image, where higher PSNR values indicate better image quality and lower distortion. PSNR is expressed in decibels (dB) and is commonly employed for objective quality assessment. PSNR is known for its speed, robustness, and ease of implementation. PSNR is calculated for the "c" component (for

example U in YUV color domain) using the following equation:

$$PSNR_c = 10 \log_{10} \frac{(2^b - 1)^2}{MSE_c} \tag{5.1}$$

Here, ”b” represents the bit-depth, and ”MSE” denotes the mean square error for component ”c” between image I and image J. The equation for calculating the mean square error (MSE) is as follows:

$$MSE_c = \frac{1}{WxH} \sum_{y=0}^{H-1} \sum_{x=0}^{W-1} (I_c^{x,y} - J_c^{x,y})^2 \tag{5.2}$$

Where ”W” stands for the width of an image, and ”H” represents the height of the image. Both ”W” and ”H” are measured in terms of pixels.

5.3.2 Weighted Spherical Peak Signal-to-Noise Ratio (WS-PSNR)

Weighted Spherical Peak Signal-to-Noise Ratio (WS-PSNR) is a quality assessment metric utilized to gauge the visual quality of images, particularly within the framework of spherical or equirectangular projections. It represents an extension of the conventional Peak Signal-to-Noise Ratio metric, specifically adapted to address the distinctive attributes of spherical images [41].

When transitioning from a two-dimensional plane to a spherical surface, a nonlinear transformation is applied. This results in a nonlinear relationship between pixels in the two planes. To accurately measure distortion in this context, individual pixel weights are assigned based on their spatial location. WS-PSNR is calculated as follows:

$$WS - PSNR = 10 \log_{10}(\frac{MAX_I^2}{WMSE}) \tag{5.3}$$

The term MAX_I represent the highest potential intensity level within the image and WMSE is weighted mean square error which incorporates weighting through the utilization of w(x,y), a weighting factor determined by a stretching ratio dependent on the specific type of projection employed. These weight values, denoted as w(x,y), play a significant role in assigning significance to the errors associated with each pixel during the computation of MSE (mean square error). This approach ensures that the impact of pixel errors on the overall MSE is proportionately adjusted based on their relative importance, in accordance with the chosen projection

type. The formula for WMSE is outlined as follows:

$$WMSE = \frac{\sum_{y=0}^{H-1} \sum_{x=0}^{W-1} (I^{x,y} - J^{x,y})^2 . w_{x,y}}{\sum_{y=0}^{H-1} \sum_{x=0}^{W-1} w_{x,y}} \tag{5.4}$$

In the given equation, H stands for the image's height, while W represents its width. The variables I and J correspond to the reference and tested images, respectively [41].

5.3.3 Immersive Video Power to Noise Ratio (IV-PSNR)

Immersive Video Power to Noise Ratio (IV-PSNR) builds upon the conventional Power to Noise Ratio metric. IV-PSNR incorporates two novel techniques, namely pixel shift and global component difference (GCD), designed to address distortions in immersive videos [42].

Pixel shift is employed in IV-PSNR to modify the traditional PSNR method, making it less sensitive to minor scene changes that might go unnoticed by viewers. For instance, if a synthesized view experiences an edge shift, this could lead to a decrease in quality when assessed using pixel value-based metrics like PSNR. However, such shifts might not be perceptible to viewers due to their lack of knowledge about the actual object edge positions in the original view. Pixel shift resolves this issue by comparing each pixel in image I with a corresponding block in image J. This difference estimation considers both the pixel in image I and the similar pixel in the collocated block of image J. This approach mitigates the problem of minor object shifts, effectively eliminating them as errors. The block size typically used in IV-PSNR is 5x5, which effectively handles shifts of less than 2 pixels [42].

Global component difference is another technique used in IV-PSNR to tackle color inconsistencies in virtual views. While color correction is often employed to reduce color artifacts in virtual views, it can sometimes result in incorrect global color characteristics. This can lead to differences when comparing two images, which may appear as errors when calculated in mathematical way but actually its barely perceptible by the viewers. GCD aims to address this issue. The global component difference is calculated for each component "c" by determining the average difference between corresponding pixels in both images. This average difference is then added to each pixel's calculation of the squared error difference [42]. The formula for calculating global component difference (GCD) is as follows:

$$GCD_c^{I \to J} = \frac{1}{W.H} \sum_{y=0}^{H-1} \sum_{x=0}^{W-1} (I_c^{x,y} - J_c^{x,y}) \tag{5.5}$$

IV-PSNR for component "c" is computed using the following equation:

$$IV - PSNR_c^{I \to J} = 10.\log_{10}(\frac{(2^b - 1)^2}{IV - MSE_c^{I \to J}}) \tag{5.6}$$

$IV - MSE_c^{I \to J}$ can be calculated using the following equation, where B represents the maximum permissible shift of the corresponding pixel:

$$IV - MSE_c^{I \to J} = \frac{1}{W.H} \sum_{y=0}^{H-1} \sum_{x=0}^{W-1} \min_{\substack{w \in [x-B, x+B] \\ h \in [y-B, y+B]}} (I_c^{x,y} - J_c^{w,h} + GCD_C^{I \to J})^2 \tag{5.7}$$

Finally, the overall value for IV-PSNR, taking into account the three components (Y, U, and V), can be calculated using the following equation:

$$IV - PSNR_{YUV}^{I \to J} = \frac{IV - PSNR_Y^{I \to J}.w_Y + IV - PSNR_U^{I \to J}.w_U + IV - PSNR_V^{I \to J}.w_V}{w_Y + w_U + w_V} \tag{5.8}$$

Here, the weight for the luma component (w_Y) is set to 4, while the weights for the chroma components (w_U and w_V) are both set to 1. These values are chosen based on the subsampling format of 4:2:0 [42].

In the case of equirectangular projection (ERP), where a spherical image is projected onto a 2D image, it's important to consider the latitude of each pixel when calculating IV-PSNR. In such scenarios, $IV - MSE_c^{I \to J}$ will be calculated as follows:

$$IV - MSE_c^{I \to J} = \frac{\sum_{y=0}^{H-1} \sum_{x=0}^{W-1} \min_{\substack{w \in [x-B, x+B] \\ h \in [y-B, y+B]}} (I_c^{x,y} - J_c^{w,h} + GCD_C^{I \to J})^2 . w_{x,y}}{\sum_{y=0}^{H-1} \sum_{x=0}^{W-1} w_{x,y}} \tag{5.9}$$

Here $w_{x,y}$ is the weight, which can be calculated as shown in equation 5.8, where AOV_V is the vertical angle of view of the ERP camera, which is equal to 180 for a full spherical camera [42].

$$w_{x,y} = \cos \frac{(y + .5 - \frac{H}{2}).\pi}{H} . \frac{AOV_v}{180^\circ} \tag{5.10}$$

5.3.4 Video Multimethod Assessment Fusion

Video Multimethod Assessment Fusion (VMAF) is a quality metric developed by Netflix to assess the perceptual quality of compressed videos. Its purpose is to predict video quality by combine various quality metrics, leveraging the strengths of each metric. VMAF employs a machine learning (ML) algorithm, specifically a support vector machine (SVM) based regres-

sion [43], to determine the final quality estimation by assigning weights to individual metrics. This ML model is trained and validated using subjective experiments involving opinion scores. VMAF integrates metrics like Visual Information Fidelity (VIF) [37] and Detail Loss Metric (DLM) [44], along with motion techniques that gauge temporal variations between neighboring frames. VMAF scores range from 0 to 100, with higher scores indicating better perceptual quality [39].

5.3.5 Objective Metrics Evaluation for TMIV

In the previous chapter, we employed TMIV software for encoding, decoding, and rendering tasks. This resulted in two types of views: the pose trace view which we can use it for subjective evaluation and the target view, which is a regenerated version of a specific input view, which we will use it in objective quality evaluation by comparing it with original view which provided as input to our encoder.

We will assess the quality of the target view using IV-PSNR software, which computes the quality using PSNR, WS-PSNR, and IV-PSNR metrics. Additionally, we'll utilize FFMPEG software (Fast Forward Moving Picture Experts Group) to calculate the VMAF metric. These metrics will guide our evaluation of the quality achieved through both the MIV anchor and Best Reference approaches.

5.3.5.1 PSNR, WS-PSNR, and IV-PSNR Calculation

In this phase, the quality assessment was conducted using the IV-PSNR software, a tool endorsed by the MPEG group. This evaluation was performed on five sequences that had been used in the implementation phase: Frog, Chess, Painter, Carpark, and ClassroomVideo sequences. IV-PSNR software had been utilized to measure PSNR, WSPSNR, and IV-PSNR.

To illustrate, here is a sample command used in the Command Prompt (cmd) to compute quality metrics for the Frog sequence in the result of the MIV anchor using IV-PSNR:

```
D:\ivpsnr-master\ivpsnr-master\buildW\Release\IV-PSNR
-i0 "D:\Workspace\Content\E01\v11_texture_1920x1080_yuv420p10le.yuv"
-i1 "D:\Workspace\Experiment\A17\E01\QP2\A17_E01_QP2_v11_tex_1920x1080_yuv420p10le.yuv"
-w 1920 -h 1080 -l 17 -bd 10 -cf 420
-o "main_miv_out.txt"
```

Here's an explanation of the parameters:

- -i0: Refers to the reference file's path, using V11 of the Frog sequence provided by CTC.

- -i1: Points to the test file's path, which is the target view "V11" generated by the TMIV decoder.
- -w: Specifies the sequence view's width (1920 pixels).
- -h: Indicates the sequence view's height (1080 pixels).
- -l: Specifies the number of frames in the provided videos (17 frames).
- -bd: Defines the bit depth of the provided videos (10-bit depth).
- -cf: Indicates the chroma format for the videos (420, representing 4:2:0 subsampling).
- -o: Specifies the output file's path where the results will be saved.

Upon executing this command, the output will include the following information:

1. PSNR values for Y, U, and V components which is our example, Y: 26.6000 dB, U: 40.7508 dB, and V: 39.4362 dB.
2. WS-PSNR values for Y, U, and V components which is our example, Y: 26.5745 dB, U: 40.7253 dB,and V: 39.4107 dB.
3. IV-PSNR value, representing an overall quality assessment which in our example 35.5827 dB.

This command allows for evaluation of video quality using the PSNR, WS-PSNR and IV-PSNR metrices, providing insights into the quality of the MIV anchor's performance in the Frog sequence. Comparable evaluations are also carried out for other sequences and the Best Reference scenario. The outcomes of these evaluations are presented in Table 5.1.

5.3.5.2 VMAF Calculation

Furthermore, to gauge the VMAF score, the FFMPEG software (Fast Forward Moving Picture Experts Group) was utilized to calculate the VMAF metric. This evaluation was performed on five sequences that had been used in the implementation phase: Frog, Chess, Painter, Carpark, and ClassroomVideo sequences, employing FFMPEG to quantify quality. To elucidate, a sample command employed in the Command Prompt (cmd) to calculate the VMAF score for the chess sequence in the context of the MIV anchor's results using FFMPEG is presented:

```
C:\Users\Mahmoud\Downloads\ffmpeg-6.0-essentials_build\ffmpeg-6.0-essentials_build\bin\ffmpeg
-video_size 2048x2048 -framerate 30 -pix_fmt
-i D:\Workspace\Experiment\A17\B02\QP2\A17_B02_QP2_v5_tex_2048x2048_yuv420p10le.yuv
-video_size 2048x2048 -framerate 30 -pix_fmt yuv420p10le
-i D:\Workspace\Content\B02\v5_texture_2048x2048_yuv420p10le.yuv
-lavfi libvmaf=model='path=model/vmaf_v0.6.1.json':log_path=vmaf_logfile.txt -f null
```

In order to calculate the VMAF score using FFMPEG, several parameters need to be provided for both the reference and tested videos. These parameters are detailed as follows:

- -video_size: This parameter specifies the resolution of the videos in the format "widthxheight." It is required for both the reference and tested videos. In our example, the resolution is set to 1920x1080.
- -framerate: The frame rate of the videos is indicated by this parameter, which need to be provided for both the reference and tested videos. In our case, the frame rate is 30 frames per second (fps).
- -pix_fmt: This parameter specifies the pixel format for the videos and is essential for both the reference and tested videos. In our example, the pixel format used is yuv420p10le.
- -i: The -i flag denotes the path to the videos that need to be tested. To ensure accurate calculations, the first index 'i' should correspond to the distorted (tested) video, while the second index is used for the reference video. For instance, in the case of the chess sequence, the target view "V5" generated by the MIV anchor is measured alongside its reference.
- -libvmaf=model: This flag is employed to specify the model used for the calculation. In our case, we provide the `'vmaf_v0.6.1.json'` model.
- -log_path: This parameter designates the path where the output results will be saved.

After executing this command, the Command Prompt (cmd) will display the VMAF score for the tested video, and the output file will contain the VMAF score for each frame. In our specific example, the obtained score is 92.2.

This command provides a means to assess video quality using the VMAF metric, offering valuable insights into the performance of the MIV anchor in the Chess sequence. Similar evaluations are conducted for the other sequences and the Best Reference scenario. The results of these evaluations are summarized in Table 5.1.

Table 5.1: Objective quality evaluation for Frog, Chess, Painter, Carpark, and ClassroomVideo sequences

Sequence Type	Operation Type	PSNR	WS-PSNR	IV-PSNR	VMAF
Frog	MIV Achor	31.08	31.06	35.6	51.5
	Best Reference	49.54	49.52	51.02	99.4
Chess	MIV Achor	45.58	45.56	49.07	92.2
	Best Reference	56.18	56.16	61.78	94.6
Painter	MIV Achor	39.52	39.49	45.07	80.93
	Best Reference	52.5	52.47	54.9	95.72
Carpark	MIV Achor	35.98	35.95	39.4	70.45
	Best Reference	55.56	55.54	57.08	97.77
ClassroomVideo	MIV Achor	38.85	38.82	41.99	77.63
	Best Reference	43.5	43.48	46.19	86.19

5.4 Conclusion

Traditionally, subjective evaluation, involving user opinions, has been the most reliable method to assess multimedia quality, including images and videos. However, it can be challenging to carry out. This has led to the rise of objective evaluation techniques, which attempt to replicate subjective assessments using mathematical approaches. This chapter delved into objective evaluation applied to the results from the previous chapter, specifically in the Frog, Chess, Painter, Carpark, and ClassroomVideo sequences under the MIV anchor and Best Reference scenarios.

Multiple metrics were employed for the evaluation process. IV-PSNR software was utilized to calculate quality metrics including PSNR, WS-PSNR, and IV-PSNR. Notably, IV-PSNR outperformed the others due to its incorporation of innovative techniques like pixel shift and global component difference, which are better suited for evaluating immersive content.

Additionally, VMAF, a cutting-edge metric, was employed. VMAF shows its effectiveness in assessing perceptual quality. VMAF combines machine learning with the strengths of various metrics to provide robust results that closely align with subjective evaluations.

In conclusion, this chapter underscored the importance of employing objective evaluation techniques to gauge the quality of compressed volumetric data. Through a combination of software tools and advanced metrics, the quality of the TMIV established by the MIV standard was examined, yielding positive outcomes. These results affirm the successful utilization of the MIV standard, enabling effective compression and subsequent decompression of input views.

Chapter 6

Conclusions

In conclusion, this thesis undertook a comprehensive investigation into the MIV standard, which is used in coding and representing immersive media. This standard's application extends to supporting virtual and extended realities, empowering users with a heightened sense of freedom. Through meticulous exploration, this research shows the the success of MIV standard to use the bit-stream effectively. MIV standard proposed TMIV shows the processes needed to encode and decode input views with high quality by implementing some smart techniques. Some of these techniques like the pruning process on the encoder side to reduce the amount of redundancy in input views, and smart synthesizers like Reference view synthesizer and View Weighting Synthesizer which are used in rendering prosses to provide the desired viewport to the viewer with a high-quality and high degree of freedom.

A dual approach, marrying theoretical understanding with practical implementation, was adopted to understand TMIV, with particular emphasis on the MIV Main Profile. The practical implementation had been done using TMIV software, while traditional 2D codecs, such as VVenC/VVdeC (H.266), were employed to encode and decode Atlases. Atlases are generated as a result of packing basic views and patches as a result of the pruning process from additional views.

Finally quality assessment by using objective quality evaluation metrics, including PSNR, WS-PSNR, IV-PSNR, and VMAF. The IV-PSNR software, a creation of the MIV standard, proved instrumental in quantifying quality aspects. Moreover, VMAF, invented by Netflix, had been used to calculate perceptual quality using FFMPEG software. This evaluation shows the ability of MIV standard to handle immersive media's representation and coding with high quality.

Looking ahead, several intriguing directions for future investigation emerge from this study. Deep learning can take place in many aspects, for example, deep learning can involve

the pruning process to prune input views in a more effective way to have high quality in the rendering process. Or use deep learning in the decoder side to estimate geometry maps especially when dealing with Geometry Absent Profile or in the case of low-quality geometry maps that had been provided as input to the encoder. The importance of geometry maps in the MIV standard comes from how they are engaged in a significant way in the whole process of compression steps and how they impact the rendering process and quality.

Bibliography

[1] Jari Takatalo, Göte Nyman, and Leif Laaksonen. Components of human experience in virtual environments. *Computers in Human Behavior*, 24(1):1–15, 2008.

[2] Ramin Ghaznavi Youvalari. *Encoding and Streaming Solutions for Immersive Virtual Reality Video*. PhD thesis, 02 2021.

[3] Martin Alain, Emin Zerman, Cagri Ozcinar, and Giuseppe Valenzise. Introduction to immersive video technologies. In *Immersive Video Technologies*. Ed. Elsevier, 2022.

[4] Ren Ng, Marc Levoy, Mathieu Brédif, Gene Duval, Mark Horowitz, and Pat Hanrahan. Light Field Photography with a Hand-held Plenoptic Camera. Research Report CSTR 2005-02, Stanford university, 2005.

[5] Bennett Wilburn, Neel Joshi, Vaibhav Vaish, Eino-Ville Talvala, Emilio Antunez, Adam Barth, Andrew Adams, Mark Horowitz, and Marc Levoy. High performance imaging using large camera arrays. In *ACM SIGGRAPH 2005 Papers*, SIGGRAPH '05, page 765–776, New York, NY, USA, 2005. Association for Computing Machinery.

[6] Edward H Adelson and John YA Wang. Single lens stereo with a plenoptic camera. *IEEE transactions on pattern analysis and machine intelligence*, 14(2):99–106, 1992.

[7] Yan Zhou, Huiwen Guo, Ruiqing Fu, Guoyuan Liang, Can Wang, and Xinyu Wu. 3d reconstruction based on light field information. In *2015 IEEE International Conference on Information and Automation*, pages 976–981, 2015.

[8] Paul Blaer and Peter Allen. Two stage view planning for large-scale site modeling. pages 814–821, 06 2006.

[9] Irene Viola, Shishir Subramanyam, Jie Li, and Pablo Cesar. On the impact of vr assessment on the quality of experience of highly realistic digital humans. 01 2022.

[10] Patrick Garus. Compression and synthesis for representation of immersive content adapted to 6dof. (compression et synthèse pour représentation de contenus immersifs adaptés au 6dof). 2022.

[11] Basel Salahieh, Joel Jung, and Adrian Dziembowski. Test model 11 for mpeg immersive video. 2021.

[12] Céline Guede, Pierre Andrivon, Jean-Eudes Marvie, Julien Ricard, Bill Redmann, and Jean-Claude Chevet. V-pcc: performance evaluation of the first mpeg point cloud codec. In *SMPTE 2020 Annual Technical Conference and Exhibition*, pages 1–27, 2020.

[13] Jill M. Boyce, Renaud Doré, Adrian Dziembowski, Julien Fleureau, Joel Jung, Bart Kroon, Basel Salahieh, Vinod Kumar Malamal Vadakital, and Lu Yu. Mpeg immersive video coding standard. *Proceedings of the IEEE*, 109(9):1521–1536, 2021.

[14] Chaofei Wang, Wenjie Zhu, Yingzhan Xu, Yiling Xu, and Le Yang. Point-voting based point cloud geometry compression. In *2021 IEEE 23rd International Workshop on Multimedia Signal Processing (MMSP)*, pages 1–5, 2021.

[15] Humberto de Jesus Ochoa Dominguez, Osslan Osiris Vergara Villegas, Vianey Guadalupe Cruz Sanchez, Efren David Gutierrez Casas, and K.R. Rao. The h.264 video coding standard. *IEEE Potentials*, 33(2):32–38, 2014.

[16] N. Minallah, S. Gul, and M.M. Bokhari. Performance analysis of h.265/hevc (high-efficiency video coding) with reference to other codecs. In *2015 13th International Conference on Frontiers of Information Technology (FIT)*, pages 216–221, 2015.

[17] Alexey Filippov and Vasily Rufitskiy. Recent advances in intra prediction for the emerging h.266/vvc video coding standard. In *2019 International Multi-Conference on Engineering, Computer and Information Sciences (SIBIRCON)*, pages 0525–0530, 2019.

[18] Gabriel Balota, Mário Saldanha, Gustavo Sanchez, Bruno Zatt, Marcelo Porto, and Luciano Agostini. Overview and quality analysis in 3d-hevc emergent video coding standard. In *2014 IEEE 5th Latin American Symposium on Circuits and Systems*, pages 1–4, 2014.

[19] Bart Kroon and Dawid Mieloch. Miv tutorial ieee vcip 2021. 2021.

[20] Vinod Kumar Malamal Vadakital, Adrian Dziembowski, Gauthier Lafruit, Franck Thudor, Gwangsoon Lee, and Patrice Rondao Alface. The mpeg immersive video standard—current status and future outlook. *IEEE MultiMedia*, 29(3):101–111, 2022.

[21] Dawid Mieloch, Patrick Garus, Marta Milovanović, Joël Jung, Jun Young Jeong, Smitha Lingadahalli Ravi, and Basel Salahieh. Overview and efficiency of decoder-side depth estimation in mpeg immersive video. *IEEE Transactions on Circuits and Systems for Video Technology*, 32(9):6360–6374, 2022.

[22] Reference view synthesizer manual. ISO/IEC JTC1/SC29/WG11 MPEG/N18068, 2018.

[23] Hanspeter Pfister Markus Gross. Point-based graphics. 2007.

[24] Adrian Dziembowski (PUT)and Bart Kroon (Philips) and Joel Jung (Tencent). Common test conditions for mpeg immersive video. 2023.

[25] Andrew Perkis, Christian Timmerer, Sabina Baraković, Jasmina Baraković Husić, Søren Bech, Sebastian Bosse, Jean Botev, Kjell Brunnström, Luis Cruz, Katrien De Moor, Andrea de Polo Saibanti, Wouter Durnez, Sebastian Egger-Lampl, Ulrich Engelke, Tiago H. Falk, Jesús Gutiérrez, Asim Hameed, Andrew Hines, Tanja Kojic, Dragan Kukolj, Eirini Liotou, Dragorad Milovanovic, Sebastian Möller, Niall Murray, Babak Naderi, Manuela Pereira, Stuart Perry, Antonio Pinheiro, Andres Pinilla, Alexander Raake, Sarvesh Rajesh Agrawal, Ulrich Reiter, Rafael Rodrigues, Raimund Schatz, Peter Schelkens, Steven Schmidt, Saeed Shafiee Sabet, Ashutosh Singla, Lea Skorin-Kapov, Mirko Suznjevic, Stefan Uhrig, Sara Vlahović, Jan-Niklas Voigt-Antons, and Saman Zadtootaghaj. Qualinet white paper on definitions of immersive media experience (imex), 2020.

[26] Kjell Brunnström, Katrien De Moor, Ann Dooms, Sebastian Egger-Lampl, Marie-Neige Garcia, Tobias Hossfeld, Satu Jumisko-Pyykkö, Christian Keimel, Chaker Larabi, Bob Lawlor, Patrick Le Callet, Sebastian Möller, Fernando Pereira, Manuela Pereira, Andrew Perkis, Antonio Pinheiro, Ulrich Reiter, Peter Reichl, Raimund Schatz, and Andrej Zgank. *Qualinet White Paper on Definitions of Quality of Experience*. 03 2013.

[27] Evangelos Alexiou, Yana Nehmé, Emin Zerman, Irene Viola, Guillaume Lavoué, Ali Ak, Aljosa Smolic, Patrick Le Callet, and Pablo Cesar. Chapter 18 - subjective and objective quality assessment for volumetric video. In Giuseppe Valenzise, Martin Alain, Emin Zerman, and Cagri Ozcinar, editors, *Immersive Video Technologies*, pages 501–552. Academic Press, 2023.

[28] Touradj Ebrahimi. Quality of multimedia experiencepast, present and future. 2009.

[29] Martin Alain, Emin Zerman, Cagri Ozcinar, and Giuseppe Valenzise. Chapter 1 - introduction to immersive video technologies. In Giuseppe Valenzise, Martin Alain, Emin Zerman, and Cagri Ozcinar, editors, *Immersive Video Technologies*, pages 3–24. Academic Press, 2023.

[30] Ji-Hwan Choe, Tae-Uk Jeong, Hyunsoo Choi, Eun-Jae Lee, Sang-Wook Lee, and Chul-Hee Lee. Subjective video quality assessment methods for multimedia applications. *Journal of Broadcast Engineering*, 12, 03 2007.

[31] T. Alpert and J. Evain. Subjective quality evaluation: the sscqe and dscqe methodologies. *EBU Technical Review*, 1997.

[32] ? Recommendation 500-10: Methodology for the subjective assessment of the quality of television pictures. ITU-R Rec. BT.500, 2000.

[33] F. Kozamernik, V. Steinmann, P. Sunna, and E. Wyckens. Samviq—a new ebu methodology for video quality evaluations in multimedia. *SMPTE Motion Imaging Journal*, 114(4):152–160, 2005.

[34] Zhou Wang, A.C. Bovik, H.R. Sheikh, and E.P. Simoncelli. Image quality assessment: from error visibility to structural similarity. *IEEE Transactions on Image Processing*, 13(4):600–612, 2004.

[35] Z. Wang, E.P. Simoncelli, and A.C. Bovik. Multiscale structural similarity for image quality assessment. In *The Thrity-Seventh Asilomar Conference on Signals, Systems Computers, 2003*, volume 2, pages 1398–1402 Vol.2, 2003.

[36] Lin Zhang, Lei Zhang, Xuanqin Mou, and David Zhang. Fsim: A feature similarity index for image quality assessment. *IEEE Transactions on Image Processing*, 20(8):2378–2386, 2011.

[37] H.R. Sheikh and A.C. Bovik. Image information and visual quality. *IEEE Transactions on Image Processing*, 15(2):430–444, 2006.

[38] H.R. Sheikh, A.C. Bovik, and G. de Veciana. An information fidelity criterion for image quality assessment using natural scene statistics. *IEEE Transactions on Image Processing*, 14(12):2117–2128, 2005.

[39] Z. Li, A. Aaron, I. Katsavounidis, A. Moorthy, and M. Manohara. Toward a practical perceptual video quality metric. 2019.

[40] M.H. Pinson and S. Wolf. A new standardized method for objectively measuring video quality. *IEEE Transactions on Broadcasting*, 50(3):312–322, 2004.

[41] Yule Sun, Ang Lu, and Lu Yu. Weighted-to-spherically-uniform quality evaluation for omnidirectional video. *IEEE Signal Processing Letters*, 24(9):1408–1412, 2017.

[42] Adrian Dziembowski, Dawid Mieloch, Jakub Stankowski, and Adam Grzelka. Ivpsnr—the objective quality metric for immersive video applications. *IEEE Transactions on Circuits and Systems for Video Technology*, 32(11):7575–7591, 2022.

[43] Corinna Cortes and Vladimir Vapnik. Support-vector networks. *Machine Learning*, 20(3):273–297, Sep 1995.

[44] Songnan Li, Fan Zhang, Lin Ma, and King Ngi Ngan. Image quality assessment by separately evaluating detail losses and additive impairments. *IEEE Transactions on Multimedia*, 13(5):935–949, 2011.