TESE DE DOUTORADO

3D IMMERSIVE CONTENT PROCESSING AND COMPRESSION

Tomás Malheiros Borges

Brasília, Novembro de 2025

UNIVERSIDADE DE BRASÍLIA

FACULDADE DE TECNOLOGIA

UNIVERSIDADE DE BRASÍLIA
Faculdade de Tecnologia

TESE DE DOUTORADO

3D IMMERSIVE CONTENT PROCESSING AND COMPRESSION

Tomás Malheiros Borges

Tese de Doutorado submetida ao Departamento de Engenharia Elétrica como requisito parcial para obtenção do grau de Doutor em Engenharia Elétrica

Banca Examinadora

Prof. Ricardo Lopes de Queiroz, Ph.D., UnB _____
Orientador (Presidente da Banca)

Prof. Eduardo Peixoto Fernandes da Silva, Ph.D., UnB _____
Examinador Interno

Profa. Carla Liberal Pagliari, Ph.D., IME _____
Examinador Externo

Prof. Luciano Volcan Agostini, Ph.D., UFPel _____
Examinador Externo

FICHA CATALOGRÁFICA

BORGES, TOMÁS MALHEIROS
3D IMMERSIVE CONTENT PROCESSING AND COMPRESSION [Distrito Federal] 2025.
xvi, 135 p., 210 x 297 mm (ENE/FT/UnB, Doutor, Engenharia Elétrica, 2025).
Tese de Doutorado - Universidade de Brasília, Faculdade de Tecnologia.
Departamento de Engenharia Elétrica

1. 3D mesh and point cloud	2. super-resolution
3. embedded coding	4. rate-distortion optimization
I. ENE/FT/UnB	II. Título (série)

REFERÊNCIA BIBLIOGRÁFICA

BORGES, T. M. (2025). *3D IMMERSIVE CONTENT PROCESSING AND COMPRESSION*. Tese de Doutorado, Publicação: PPGEE 216/25, Departamento de Engenharia Elétrica, Universidade de Brasília, Brasília, DF, 135 p.

Tomás Malheiros Borges
Depto. de Engenharia Elétrica (ENE) - FT
Universidade de Brasília (UnB)
Campus Darcy Ribeiro
CEP 70919-970 - Brasília - DF - Brasil

Acknowledgments

This thesis bears only my name on the cover, but it represents the collective effort, guidance, and friendship of many people who supported me throughout this journey. I am deeply grateful to all of them.

To my advisor, Prof. Ricardo Queiroz, I owe my deepest gratitude for his guidance, insight, and encouragement from the very beginning of this work. His thoughtful feedback continually challenged me to refine my ideas and pursue rigor and clarity at every step.

To my examining committee, Profs. Eduardo Peixoto, Carla Pagliari, and Luciano Agostini, I extend my sincere thanks for their time, constructive comments, and valuable suggestions that greatly improved the quality of this thesis.

To my colleagues from the DIVP group, I am grateful for their collaboration and support. I am especially thankful to Prof. Diogo Garcia for providing the tools that made the super-resolution work possible, to Dr.-to-be Victor Fabre for his constant help, discussions, and encouragement, and to Dr. Davi Freitas for his valuable insights and enthusiasm.

To my former colleagues from InterDigital, I express my appreciation for their companionship and mentorship. I am especially grateful to Dra. Maja Krivokuća, whose collaboration and friendship were essential to this work and continue to inspire me. I also thank Dr. Olivier Roupin for patiently helping me with C++, and to my dear friend Dr. Gustavo Sandri for his unwavering support throughout this journey.

To my colleagues from Fraunhofer HHI, I extend my thanks for welcoming me into their team and for the stimulating environment they provided. I am especially thankful to Dr. Yago Sánchez, whose guidance, collaboration, and generosity greatly enriched this work, to Robert Skupin for our productive discussions and for kindly sharing his desk with me, and to Dr. Cornelius Hellge and Dr. Thomas Schierl for their continued support and encouragement.

To my friends, I am thankful for the joy, balance and perspective they brought to my life beyond research.

To my family, I owe my deepest appreciation for their unconditional love and support. Their patience and understanding sustained me through every challenge, and this achievement would not have been possible without them.

The assistance of the PPGEE department staff in helping me navigate the inevitable bureaucratic hurdles throughout this journey is also gratefully acknowledged. This work was partially supported by CNPq/CAPES under grant 88887.600000/2021-00.

ABSTRACT

This thesis focuses on advancing compression and super-resolution techniques for 3D point clouds and dynamic meshes, pivotal in applications such as virtual reality, autonomous navigation, and telepresence. The work comprises several contributions, each targeting a specific challenge within these domains. Initially, we present a novel fractional super-resolution method designed to enhance the geometry of voxelized point clouds, improving the quality of lossy compression in the Moving Picture Experts Group's (MPEG) Geometry-based Point Cloud Compression (G-PCC). By leveraging neighborhood self-similarities, our approach enables detailed geometry restoration from downsampled inputs without the need for extensive training, achieving results comparable to machine-learning techniques.

The thesis also explores a zerotree coding approach within MPEG's Video-based Dynamic Mesh Coding (V-DMC) framework for time-varying meshes (TVMs). Utilizing an edge-based hierarchical tree structure, this method provides progressive, quality-scalable encoding for subdivision wavelet coefficients, achieving efficient compression for complex 3D mesh sequences.

Finally, we propose an importance-weighted bit allocation scheme for optimizing 3D asset compression in immersive viewpoint applications. By modeling distortion based on quantization parameters, position, and camera factors, our approach improves visual fidelity in key areas without exceeding bitrate constraints.

Together, these contributions address the growing demand for efficient, high-quality 3D data compression in immersive applications, supporting the development of adaptable and resource-efficient solutions for diverse real-world scenarios.

RESUMO

PROCESSAMENTO E COMPRESSÃO DE CONTEÚDOS 3D IMERSIVOS

Esta tese aborda avanços em técnicas de compressão e super-resolução para nuvens de pontos e *meshes* dinâmicas em 3D, representações 3D fundamentais em aplicações como realidade virtual, navegação autônoma e telepresença. O trabalho inclui várias contribuições, cada uma voltada para desafios específicos dentro desses domínios.

Inicialmente, apresentamos um método inovador de super-resolução fracionária para aprimorar a geometria de nuvens de pontos voxelizadas, elevando a qualidade da compressão com perdas do codificador *Geometry-based Point Cloud Compression* (G-PCC), desenvolvido pelo *Moving Picture Experts Group* (MPEG). Ao explorar auto-similaridades locais, nossa abordagem permite a reconstrução detalhada da geometria a partir de entradas com amostragem reduzida, sem a necessidade de treinamento intensivo, e alcança resultados comparáveis aos obtidos por técnicas baseadas em aprendizado de máquina.

A tese também investiga uma abordagem de codificação baseada em *zerotrees* no contexto do *Video-based Dynamic Mesh Coding* (V-DMC), desenvolvido pelo MPEG, voltado para *meshes* dinâmicas com variabilidade temporal. Utilizando uma estrutura hierárquica em árvore, construída a partir das arestas dos triângulos, o método permite codificação escalonável em qualidade e alcança compressão eficiente para sequências de *meshes* complexas.

Por fim, propomos uma estratégia de alocação de bits que considera a importância perceptual de cada ativo tridimensional com base no ponto de vista do usuário. O objetivo é otimizar a distribuição de bits entre múltiplos ativos 3D em aplicações imersivas, priorizando os elementos mais relevantes para a experiência visual. Essa abordagem permite uma otimização do *trade-off* taxa-distorção guiada pela relevância perceptual da perspectiva do observador. Os resultados experimentais demonstram que a estratégia proposta proporciona economias significativas de taxa de bits e melhora a qualidade da experiência do usuário em comparação com a alocação uniforme, sendo sua eficácia dependente da disposição dos ativos na cena e do movimento da câmera.

Coletivamente, essas contribuições respondem à crescente demanda por compressão eficiente e de alta qualidade de dados 3D em aplicações imersivas, promovendo o desenvolvimento de soluções adaptáveis e com uso eficiente de recursos para diversos cenários reais.

CONTENTS

LIST OF FIGURES

LIST OF TABLES

Relevant Equations

$V = \{\mathbf{v}(k)\},$ with $\mathbf{v}(k) = (x_k, y_k, z_k)$	Definition of the list of occupied voxels.
$V_d = \texttt{unique}(\texttt{round}(V/s))$	Definition of the grid downsampling.

Symbols

V	The geometry of a point cloud, a list of 3D positions, i.e. occupied voxels.
C	List of colors associated with the occupied voxels from V.
N	List of normal vectors associated with the occupied voxels from V.
$\mathbf{v}(k)$	Vector representing an occupied voxel from V

Subscripts (Chapter 3)

d	Indicative of downsampling.
d^2	Indicative of downsampling from a previously downsampled input.
u	Indicative of NNI upsampling.
e	Indicative of upsampling by simple expansion.
sr	Indicative of the proposed super-resolution method.
LS	Indicative of usage of the Laplacian smoothing.

Definitions

L_1 distance	Also known as city block distance, or Manhattan distance, it is the sum of lengths of the projections of the line segment between the points onto the coordinate axes. For vectors $\mathbf{a} = (a_1, a_2, a_3)$ and $\mathbf{b} = (b_1, b_2, b_3)$ $$\|\mathbf{a} - \mathbf{b}\|_1 = \sum_{i=1}^{3} \|a_i - b_i\|.$$
D1 metric	Point-to-point metric. Equal to $\max\{D1_O, D1_D\}$, where the first measurement relates to the omission of correct points and the second one with the excess of incorrect (extra) points.
D2 metric	Point-to-plane metric. Measures perceived surface distortions. It is the D1 metric projected along the normal direction of each evaluated point.
octree	A tree structure in which each internal node has exactly eight children.
pixel	Picture element.
predlift	Combined Predicting-Lifting Transform used in G-PCC.

trisoup	Surface approximation method used in G-PCC.
voxel	Volume element.

Acronyms

2D	Two-dimensions
3D	Three-dimensions
3DGH	Coding of 3D Graphics and Haptics
6DoF	Six Degrees of Freedom
AC	Arithmetic Coding
AFX	Animation Framework eXtension
AI	Artificial Intelligence
AI-PCC	Artificial-Intelligence-based Point Cloud Compression
AR	Augmented Reality
ASCII	American Standard Code for Information Interchange
AVC	Advanced Video Coding
BD	Bjøntegaard Delta
bpif	Bits per input face
bpiv	Bits per input voxel
CABAC	Context-Adaptive Binary Arithmetic Coding
CfP	Call for Proposals
CIE	Commission Internationale d'Éclairage
CTC	Common Test Conditions
DASH	Dynamic Adaptive Streaming over HTTP
DCT	Discrete Cosine Transform
DIS	Draft International Standard
DWT	Discrete Wavelet Transform
E-G-PCC	Enhanced G-PCC
EBCOT	Embedded Block-Coding with Optimized Truncation
EZW	Embedded Zerotree Wavelet
FAMC	Frame-based Animated Mesh Compression
fps	Frames per second
FTV	Free-Viewpoint Television
G-PCC	Geometry-based Point Cloud Compression
GIS	Geographic Information Systems
GPU	Graphics Processing Unit
GSC	Gaussian Splat Coding
GSM	Gaussian Scale Mixture
GTV	Graph Total Variation

HEVC	High Efficiency Video Encoding
HVS	Human Visual System
IBSM	Image-based Sampling Metric
ICIP	International Conference on Image Processing
IDCM	Inferred Direct Coding Mode
IEC	International Electrotechnical Commission
IEEE	Institute of Electrical and Electronics Engineers
IQR	Interquartile Range
ISO	International Organization for Standardization
ITU-T	International Telecommunication Union – Telecommunication Standardization Sector
IZ	Isolated Zero
JPEG	Joint Photographic Experts Group
JTC1	Joint Technical Committee 1 – Information Technology
JVET	Joint Video Experts Team (of ITU-T's VCEG and ISO/IEC's MPEG)
LiDAR	Light Detection And Ranging sensor
LoD	Level of Detail
LR	Low-resolution
LS	Laplacian Smoothing
LUT	Look-up table
MEB	MPEG Edgebreaker
ML	Machine Learning
MLS	Mean Lest Squares
MPEG	Moving Pictures Experts Group
MSE	Mean-Squared Error
MSSIM	Mean SSIM
MVUB	Microsoft Voxelized Upper Body
NEG	Significant and Negative Coefficient
NNI	Nearest Neighbor Interpolation
OBJ	Wavefront Object File
OBUF	Optimal Binarization with Update on-the-Fly
PCC	Point Cloud Compression
PLY	Stanford Polygon File Format
PMSE	Projected MSE
PNG	Portable Network Graphics
POS	Significant and Positive Coefficient
PPSNR	Projected PSNR
PSNR	Peak Signal-to-Noise Ratio
PSSIM	Projected SSIM
PVIFp	Projected VIFp
QoE	Quality of Experience

QP	Quantization Parameter
RAHT	Region-Adaptive Hierarchical Transform
RD	Rate-Distortion
RDO	Rate-Distortion Optimization
RGB	Red, Blue and Green color space
RLGR	Run-Length Golomb-Rice
ROI	Region-of-interest
S-G-PCC	Solid Geometry Point Cloud Compression
SC29	Sub Committee 29 – Coding of Audio, Picture, Multimedia and Hyper-media Information
SHEVC	Scalable Extensions of the HEVC
SOT	Spatial Orientation Tree
SPIHT	Set Partitioning in Hierarchical Trees
SR	Super-resolution
SSIM	Structural Similarity Index Measure
SVC	Scalable Video Coding Extension of the AVC
TIP	Transactions on Image Processing
TM	Test Model
TMC13	Test Model Categories 1 and 3 – G-PCC
TMC2	Test Model Category 2 – V-PCC
TVM	Time-Varying Meshes
V-DMC	Video-based Dynamic Mesh Compression
V-PCC	Video-based Point Cloud Compression
V3C	Visual Volumetric Video-based Coding
VCEG	Video Coding Experts Group
VIF	Visual Information Fidelity
VIFp	VIF in pixel domain
VR	Virtual Reality
VVC	Versatile Video Coding
WD	Working Draft
WG4	Working Group 4 – MPEG Video Coding
WG7	Working Group 7 – MPEG 3DGH
WG11	Working Group 11 – Coding of Moving Pictures and Audio (MPEG)
XR	Extended Reality
YCbCr	Luma, chrominance blue, and chrominance red color space
ZT	Zerotree
ZTR	Zerotree Root

1 INTRODUCTION

1.1 CONTEXT AND MOTIVATION

The significant evolution of extended reality (XR) technologies, including handheld capturing devices, professional volumetric studios, tablets and headsets, has driven the growing popularity of immersive applications, such as augmented and virtual reality (AR and VR), autonomous navigation, telepresence, and free-viewpoint television (FTV) [1]–[5]. As a result, there has been a growing demand for efficient 3D representation formats capable of handling the complexity and scale of this content. This rising interest has not only fostered research in both academia and industry but also motivated standardization bodies such as the Joint Photographic Experts Group (JPEG) and the Moving Picture Experts Group (MPEG) [6]–[8].

The WG7 – MPEG Coding of 3D Graphics and Haptics (3DGH)[1] has been working on standards for 3D content, including point clouds and dynamic meshes. At first, the group worked on two major point cloud compression (PCC) standards: geometry-based PCC (G-PCC) and video-based PCC (V-PCC) [9], [10], focused on static and dynamic point clouds, respectively. Later, the group started looking for compression solutions for dynamic time-varying meshes (TVMs) [11]. Nevertheless, effective compression of 3D graphics remains an active research problem, especially when considering real-time streaming, bandwidth limitations, and computational constraints.

This thesis is developed within this evolving landscape, contributing to ongoing advancements while addressing key gaps that current solutions have yet to resolve. By focusing on improving compression efficiency, enhancing reconstruction techniques, and optimizing bit-allocation strategies, this work seeks to provide meaningful contributions toward advancing the state of 3D content representation and transmission.

1.2 CHALLENGES AND CONTRIBUTIONS

The efficient representation and compression of 3D content remains a fundamental challenge, particularly as immersive applications demand higher quality, lower latency, and greater scalability. Although significant advancements have been made in point cloud and mesh coding, several open problems persist.

This thesis addresses three key challenges:

[1] https://www.mpeg.org/structure/coding-of-3d-graphics/

1. Super-resolution for voxelized point clouds

    - **Problem Statement:** Existing PCC methods often rely on downsampling voxelized point clouds to reduce their size, leading to information loss and quality degradation. However, conventional super-resolution techniques are not well-suited for this scenario, as they fail to account for voxelization constraints and the need for fractional scaling.
    - **Contributions:** Building upon the research conducted during the Master's program [12], this work proposes a fractional super-resolution method tailored to voxelized point clouds and grid downsampling, enhancing the quality of compressed content by leveraging spatial self-similarities.
    - **Outcome:** The method was successfully integrated into PCC pipelines, leading to journal publications [13], [14], and contributions to MPEG 3DGH [15]–[17].

2. Scalable geometry coding for dynamic TVMs

    - **Problem Statement:** The current state-of-the-art TVM codec, MPEG V-DMC, encodes geometry by packing subdivision wavelet coefficients into 2D images, which are then compressed using standard video codecs. While effective, this approach does not fully exploit the 3D hierarchical nature of wavelets and lacks quality scalability.
    - **Contributions:** This work proposes a zerotree-based coding method that directly encodes subdivision wavelet coefficients in the 3D space, rather than packing them into 2D frames. This enables fully progressive decoding of the mesh geometry, adding the desired support for quality scalability.
    - **Outcome:** This research led to an IEEE TIP publication [18] (also invited for presentation at the IEEE ICIP 2025 in Alaska, USA), a patent application [19], and several technical contributions to MPEG 3DGH [20]–[23].

3. Rate-distortion optimization for immersive applications

    - **Problem Statement:** In immersive-viewpoint applications, multiple 3D assets must be transmitted efficiently, requiring careful tuning of compression parameters to balance bitrate and visual quality. Existent rate-distortion optimization methods often do not consider the joint allocation of bits across multiple assets in a scene based on viewpoint relevance, leading to inefficient resource distribution and suboptimal perceived quality.
    - **Contributions:** A rate-distortion optimization model was developed to intelligently allocate bitrates across multiple 3D assets, optimizing perceived quality within bandwidth constraints. The model considers the visual importance of each asset and the impact of bitrate allocation on the overall quality of the scene.
    - **Outcome:** This research forms the basis of a manuscript submitted for publication.

These contributions stem from research conducted across three institutions—UnB (Brasília, Brazil), InterDigital (Rennes, France), and Fraunhofer HHI (Berlin, Germany)—each influencing

the focus and scope of the work. While the topics may seem distinct, they are interconnected by a common goal: advancing the efficiency and flexibility of 3D content compression to support next-generation immersive applications.

1.3 OBJECTIVES

The objectives of this work are as follows:

1. To review and analyze the fundamental challenges in 3D content representation and compression, focusing on point clouds and meshes as key formats for immersive applications. This includes an in-depth study of MPEG's PCC and TVM compression frameworks, scalable coding techniques, 3D quality metrics, and rate-distortion theory, providing the necessary foundation for the proposed contributions.
2. To investigate and develop a fractional super-resolution method for voxelized point clouds that meets the requirements of PCC, particularly ensuring compatibility with voxelized representations and supporting fractional scale factors.
3. To propose and implement a scalable coding approach for the geometry of dynamic TVMs, leveraging wavelet zerotree coding to improve compression efficiency while enabling embedded, progressive decoding.
4. To design and validate a rate-distortion optimization model that allocates bits across multiple 3D assets in immersive-viewpoint applications, ensuring efficient use of bandwidth and improved visual quality.
5. To contribute to standardization efforts, particularly within MPEG 3DGH, by providing technical insights, experimental evaluations, and reports that support the integration of the developed techniques into practical coding frameworks.

1.4 MANUSCRIPT PRESENTATION

This thesis begins with an overview of the fundamental concepts of 3D content representation and compression in Chapter 2, providing the necessary background for understanding the subsequent chapters. Next, in Chapter 3, the fractional super-resolution method is presented, along with an additional related contribution that emerged as extension of this work. In Chapter 4, the proposed method for coding the geometry of dynamic mesh coding is presented. Then, in Chapter 5, the model for bit allocation in immersive applications is presented. Finally, Chapter 6 summarizes the contributions, discusses their implications, and outlines directions for future research.

2 BACKGROUND & FUNDAMENTALS

This chapter provides the foundations required for the work presented in the subsequent chapters. It first reviews common 3D representations (point clouds and meshes) and the basic concepts of volumetric sampling and visualization that underpin rendering and quality assessment. Next, it introduces rate-distortion principles, performance metrics, and objective measures used for 3D content evaluation, paying particular attention to point-based and projection-based metrics adopted by MPEG 3DGH. The chapter also reviews prevailing compression approaches for 3D content, with focused overviews of the geometry-based point cloud codec and the video-based dynamic mesh codec, and concludes by recalling key aspects of scalable coding.

2.1 3D REPRESENTATIONS: POINT CLOUDS AND MESHES

Throughout this thesis, we work primarily with two explicit 3D data formats: voxelized point clouds and polygonal meshes. Next, we introduce the notation and practical properties of these representations, which will be used later when discussing rendering, compression, and quality assessment. Other representations (e.g., 3D Gaussian splats, implicit surfaces, signed-distance fields, or learned neural scene representations) provide useful trade-offs in rendering and compression, but they fall outside the scope of this work and are not treated here.

2.1.1 POINT CLOUDS

Representing 3D objects requires spatial sampling of their surface or volume, where each sample captures specific properties at a defined 3D position (x, y, z). These samples are referred to as *voxels* (volume elements), and the properties associated with them are called *attributes*, which can indicate a variety of things, such as color, reflectance, normals, opacity, material labels, etc. Voxels can simply differentiate background from the presence of an object in space or can assume more complex roles like density, heat, pressure, color, among other properties, or a combination of them depending on the application. When volumetric data is acquired, e.g., through 3D surface scanning of complex object surfaces, a scattered 3D point set, also called a *point cloud* is obtained [24], [25]. An example of such 3D captured object in its point cloud format is shown in Figure 2.1.

Point cloud data, thus, comprise a list V containing K occupied voxels, representing sampled points from a 3D surface. For the applications of this work, the main attribute considered is a list

Figure 2.1: The Stanford Bunny. One of the most commonly used 3D models in computer graphics, which was captured using a range scanner [26], [27].

C of sampled colors, and sometimes a list N of surface normals. The following notation is used:

$$V = \{\mathbf{v}(k)\}, \text{ with } \mathbf{v}(k) = (x_k, y_k, z_k), \tag{2.1}$$

$$C = \{\mathbf{c}(k)\}, \text{ with } \mathbf{c}(k) = (R_k, G_k, B_k), \tag{2.2}$$

$$N = \{\mathbf{n}(k)\}, \text{ with } \mathbf{n}(k) = (n_{x_k}, n_{y_k}, n_{z_k}), \tag{2.3}$$

for $k = 1, 2, \ldots, K$. Notice that, although V does not need to be sorted in any particular order, the attributes C and N must be sorted in the same order as V.

For efficient processing, point clouds are usually quantized in a cubic grid. Thus, henceforth we narrow the voxel definition to a *quantized* volume element. For example, in an integer-defined 3D grid, a voxel will be any of the $1 \times 1 \times 1$ cubes forming the grid. This quantization process is known as *voxelization*, and point clouds quantized in this way are called voxelized point clouds.

2.1.2 Meshes

Point clouds are a natural output of 3D acquisition devices, as they directly reflect the discrete sampling performed by sensors. Their simplicity offers a good compromise between capturing the subject with reasonable quality and maintaining a low reconstruction (or rendering) cost. However, because point clouds do not explicitly represent surface connectivity, producing high-resolution renderings becomes more challenging. Accurately conveying fine surface details and achieving a *watertight* representation, i.e., without holes or gaps in the surface, typically requires a high number of points. While splatting techniques can help reduce the point count by blending neighboring samples, they often come with increased computational cost [28].

A more structured alternative is to represent 3D objects using *meshes*. In a mesh, *vertices* define the points in 3D space, *edges* connect these vertices, and *faces* span the surface enclosed by the edges. This additional connectivity information allows for an explicit representation of the object's *topology*, as illustrated in the wireframe model in Figure 2.2.

Polygon meshes approximate a smooth surfaces by a collection of planar polygons (usually trian-

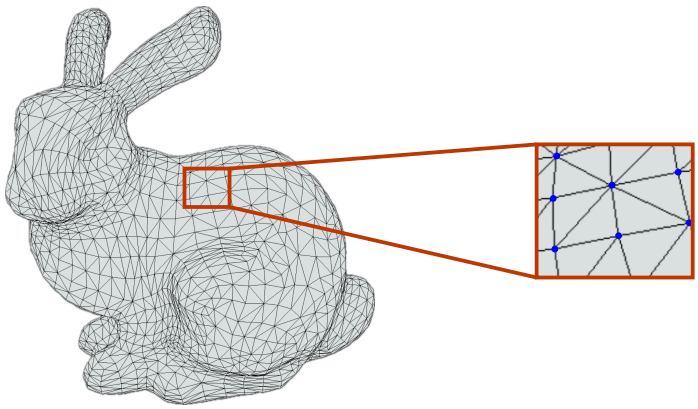


Figure 2.2: Mesh wireframe model. The vertices are represented by the blue dots, the edges by the black lines, and the faces by the gray triangles.

gles). However, accurately capturing fine details often demands a high number of polygons, which can significantly increase the data size. This makes efficient mesh compression essential for enabling the use, storage, and transmission of detailed 3D models. Nevertheless, polygon meshes remain a versatile and efficient representation. They can model complex topologies, are easy to derive from other surface representations, and integrate well into 3D processing pipelines. Their geometric flexibility makes them a dominant choice for representing 3D shapes in various applications [29].

2.1.2.1 Mesh Fundamentals

Vertex degree (or valence) is the number of vertices incident to that vertex. A mesh is said to have *regular* connectivity if all its vertices have the same degree (usually 6 for a triangle mesh), *irregular* connectivity otherwise, and *semi-regular* if all but a few extraordinary vertices have different degrees. The different types of connectivity are illustrated in Figure 2.3.

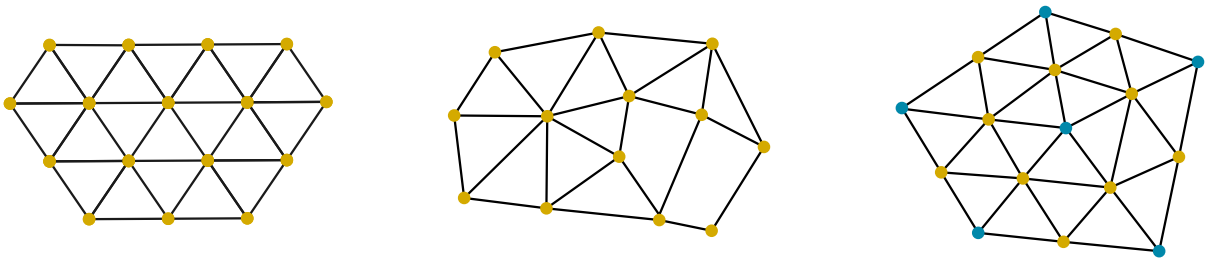


Figure 2.3: Mesh connectivity types. Regular connectivity in (a), irregular connectivity in (b), and semi-regular connectivity in (c) [30].

In practice, most meshes exhibit irregular connectivity. However, they are often remeshed into regular or semi-regular structures to simplify processing and improve compression efficiency. A mesh is considered open if it contains *boundary edges*, i.e., edges that are connected to only one face instead of two. The presence of such boundaries typically imposes at least a semi-regular structure on the mesh.

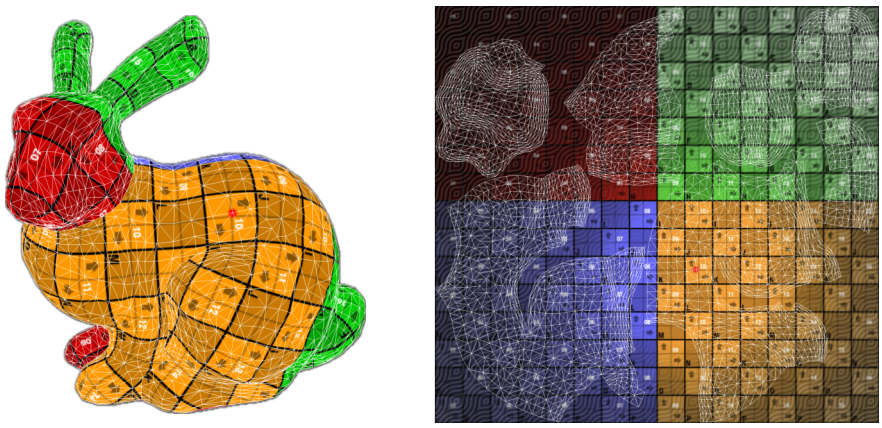
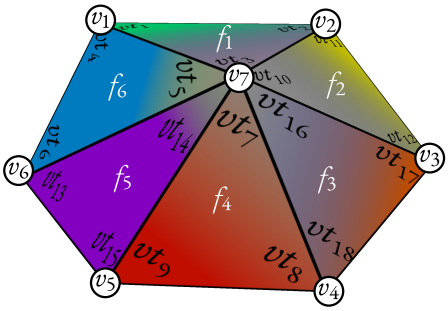


Figure 2.4: UV mapping illustration using checkerboard pattern applied to the *bunny*. The UV coordinates are used to map the texture onto the mesh surface, allowing for detailed photometric representation.

Attributes in a mesh can be associated either per vertex (as with point clouds) or per face. Common attributes include RGB color (or texture) and surface normals. When color is assigned per vertex, it is typically linearly interpolated across the faces, resulting in smooth shading and gradual color transitions. Alternatively, assigning texture per face allows for high photometric accuracy, without requiring high geometric precision.

In textured mesh representations, a dense mesh is often simplified (i.e., decimated) to produce a lower-resolution mesh that approximates the original surface using fewer triangles, representing the original surface by piecewise linear approximation. The vertex colors of the dense mesh are then projected into one or more 2D texture maps, and (u, v) coordinates are assigned to the vertices of the simplified mesh. These coordinates define how the 2D texture maps are applied to the 3D mesh surface. This mapping, commonly referred to as UV mapping, is illustrated in Figure 2.4, where a checkerboard pattern is applied to a mesh. At rendering time, the simplified mesh, along with its UV coordinates and texture maps, allow each triangle to be filled with detailed photometric content extracted from the texture. This method decouples geometric complexity from visual richness, allowing even a coarse geometry to be rendered with detailed surface appearance. In addition, UV mapping supports efficient compression of the texture as the images can be encoded using standard 2D image formats (e.g., PNG, JPEG) and sequence of texture maps can be compressed with video codecs like High Efficiency Video Coding (HEVC) [31] or Versatile Video Coding (VVC) [32]. Owing to these advantages, along with ease of editing and integration into existing graphics pipelines, the textured mesh has become the most widely adopted representation for 3D models. Its popularity is further reinforced by the fact that modern GPUs are specifically optimized for rendering textured meshes [29].

A common way of representing a mesh is through an *indexed face set*, which is the base of most ASCII file formats for meshes such as OFF (Object File Format), OBJ (Wavefront Object), and PLY (Polygon File Format) [33]. In this representation, the mesh is defined by a geometry array containing floating-points vertex coordinates, and a connectivity array specifying the indices of the vertices that form each face. Attribute representation vary among file types and depends on both

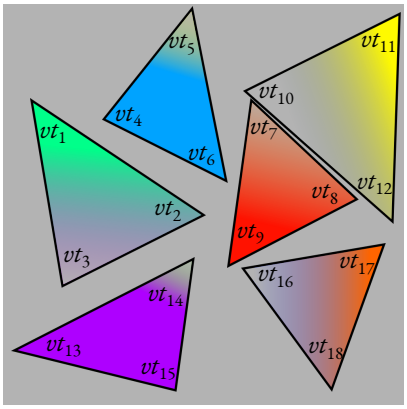


(a)

```
1  mtlib mesh.mtl
2  # List 3D vertices (x, y, z) (float)
3  v  x1   y1   z1
4  v  x2   y2   z2
5  v  x3   y3   z3
6  v  x4   y4   z4
7  v  x5   y5   z5
8  v  x6   y6   z6
9  v  x7   y7   z7
10
11 # List of texture coordinates (u, v) (float)
12 vt u1   v1
13 vt u2   v2
14 vt u3   v3
15 vt u4   v4
16 vt u5   v5
17 vt u6   v6
18 vt u7   v7
19 vt u8   v8
20 vt u9   v9
21 vt u10  v10
22 vt u11  v11
23 vt u12  v12
24 vt u13  v13
25 vt u14  v14
26 vt u15  v15
27 vt u16  v16
28 vt u17  v17
29 vt u18  v18
30
31 # List of faces (vertex/texture) (uint)
32 usemtl material0
33 f  1/1  2/2  7/3
34 f  2/11 3/12 7/10
35 f  3/17 4/18 7/16
36 f  4/8  5/9  7/7
37 f  5/15 6/13 7/14
38 f  6/6  1/4  7/5
```

mesh.obj

(b)



texture_map.png

(c)

```
1  newmtl material0
2  map_Kd texture_map.png
```

mesh.mtl

(d)

Figure 2.5: Example of an indexed face set representation of a mesh using the OBJ syntax. (a) Triangular mesh composed of 7 vertices and 6 faces; (b) Corresponding OBJ file syntax: line 1 defines the material library file, followed by the geometry array (lines 3–9), UV coordinates (lines 12–29), and the face connectivity (lines 33–38). Line 32 specifies the material applied to the subsequent faces; (c) Texture map associated with the UV coordinates; (d) MTL file syntax defining the material properties, including the path to the texture map.

the type of attribute and how it is defined. In the case of the OBJ format for textured meshes, attributes are organized into independent arrays, which are then cross-referenced by the mesh faces through their own index lists, such as:

- Surface normals (vn) are stored in a separate 3D array of floating-point vectors and indexed per face corner, they are useful for shading and lighting calculations;
- UV coordinates (vt) are stored in a separate 2D array of floating-point vectors and indexed per face corner, they are used to map textures onto the mesh surface;
- Material properties can be defined in an external MTL file, referenced from within the OBJ file. This file specifies surface properties such as diffuse and specular reflectance, shininess, and texture maps. Different faces can reference different materials using the `usemtl` keyword.

This flexible structure allows OBJ files to support detailed surface appearance while remaining relatively simple and widely compatible. Figure 2.5 provides an example of this structure, illustrating

how geometry, UV coordinates, and material properties are represented and referenced within an OBJ textured mesh file.

Although straightforward to generate and interpret, the indexed face set is far from an efficient representation for triangle meshes. This representation stores geometry, connectivity, and attribute data explicitly, often leading to significant redundancy. For instance, we can see from Figure 2.5(b) (l. 33–38), that all the vertex indices of in face definitions appear at least twice, and vertex 7 appears six times, because each vertex is shared among several faces and must be re-referenced for each one. In large meshes, especially those with rich attribute sets (e.g., texture coordinates, normals), this redundancy can result in files that are large, slow to parse, and inefficient to store or transmit. Furthermore, as illustrated in Figures 2.5(b) and (c), a poorly designed UV map may require a large number of texture coordinates to avoid texture distortion or visible seams. Constructing an efficient and seamless UV mapping often demands careful alignment between texture layout and mesh topology, which is a non-trivial task in practice.

Efficient mesh compression techniques aim to eliminate redundancy by exploiting geometric regularities and topological coherence, significantly reducing the storage and transmission costs. These methods are crucial not only for scalable rendering and storage but also for enabling interactive applications and real-time streaming of complex 3D content. In Section 2.5.2, we explore some of such techniques.

2.2 VOLUME VISUALIZATION

Depth perception comes from both monocular and binocular cues from the human visual system (HVS). Binocular vision allows for *stereopsis*, or stereo vision, which is the visual brain's ability to register a sense of 3D from visual inputs coming from both eyes [34], [35]. In stereo vision, each eye sees a slightly different version of the scene, which is projected to the retinas as 2D images, and then the brain extrapolates depth. Monocular vision perceives depth by information outside the 3D volume itself. It uses cues such as motion, perspective, lighting, shading, and depth of field [35]. Thus, to visualize volumetric data, we first need to find a way to project a 3D volume into a 2D screen. Then, to give the depth perception, we need to add some depth cues to the scene.

Volume Visualization is the field of Computer Science that targets this problem. Its main role is to create images that convey various insights about the input volumetric data [25]. Possible solutions vary depending on user/application requirements and computational restrains. Nonetheless, all such solutions follow the same conceptual framework called the volume visualization pipeline [25], [36], as depicted in Figure 2.6.

First, raw volumetric data is acquired. It may be by sampling 3D data, or by modeling analytical data. Then, there is a filter step to enhance data, where operations such as selecting only a subset of the imported dataset, outlier removal, coordinate transformation, filtering, resampling, or quantization are performed. The next step is to map abstract data to visual representation. This can

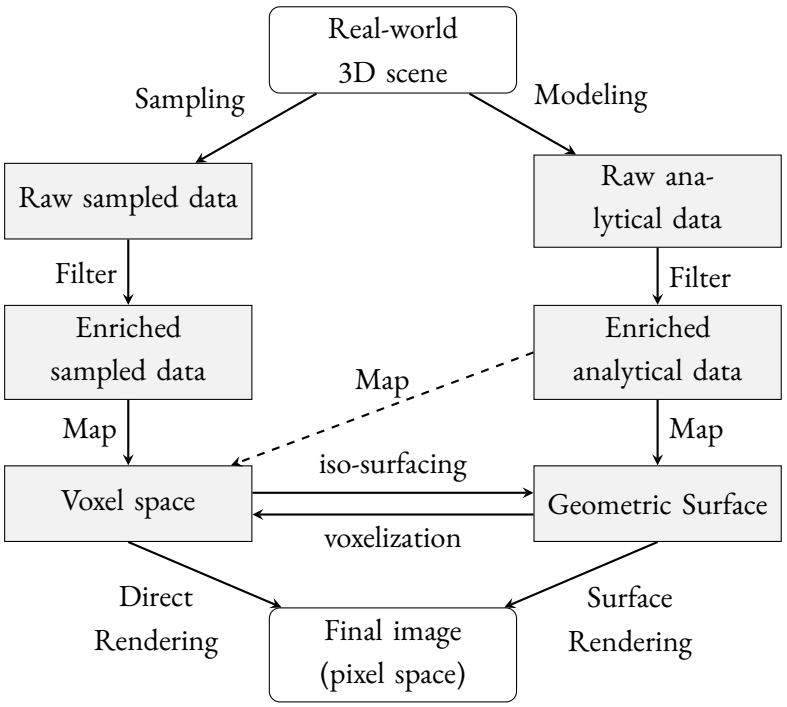


Figure 2.6: The Volume Visualization Pipeline.

be achieved either in the voxel space (i.e., discrete 3D points) or by leveraging interpolations and connectivity information to construct a geometric surface representation, such as a polygon mesh. Transforming from voxels to a surface-based geometry is possible using iso-surfacing algorithms (e.g., marching cubes [37], [38], or more recente approaches [39], [40]). The inverse transformation is also possible by voxelizing the geometry surface in discrete and quantized points. Sometimes, although rarely, analytical data is directly mapped to the voxel space. The final process is the rendering step, in which the mapped 3D data is rendered, or "drawn" into the 2D screen, together with the user-specified viewing parameters such as viewpoint and lighting [25]. There are many rendering algorithms, and they are usually divided amongst direct and indirect methods. Direct rendering methods render the volumetric data directly from voxels to pixels, while indirect methods, like surface rendering, require an additional interpolation method. Indirect methods usually have better-perceived quality, especially when few volumetric samples are available. However, there is an added computational cost and storage. For real-time applications, direct methods can provide a faster and more straightforward, albeit less accurate, implementation [25].

2.2.1 Virtual Camera

Before imaging the mapped volumetric data, a virtual camera must be set up in order to provide both a point of view and the necessary parameters for projecting the 3D volume in a 2D plane. A virtual camera is specified by its extrinsic and intrinsic parameters. The extrinsic parameters include the camera position (eye e), the point it is looking at (center c), and the up direction vector $\mathbf{u}$. Together they describe the camera's location and orientation in space. The intrinsic parameters

define how the projection is performed, including the vertical field-of-view angle ϕ_{fov}, the aspect ratio w/h, and the distances to the near and far clipping planes (z_{near} and z_{far}), which define the depth range of the scene that is rendered. These parameters define the view volume in a pyramid frustum shape, also called the view frustum, as depicted in Figure 2.7. Only objects inside this frustum are considered visible by the camera and rendered in the final image.

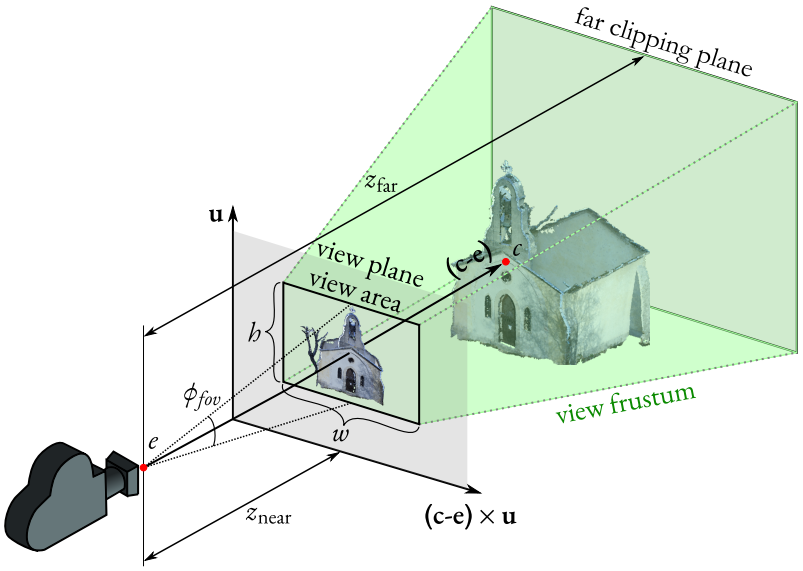
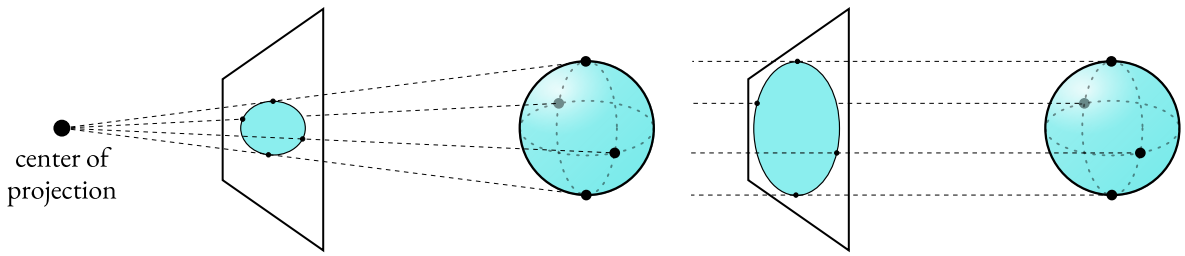


Figure 2.7: The virtual camera parameters [25].

The virtual camera works similarly to a real camera by capturing a perspective projection in which all light rays coming from the 3D scene converge at a common point. The image is formed on a planar surface between the center of projection and the 3D scene. In perspective projections, objects further away from the view plane appear smaller, an effect called foreshortening. Although this is the more natural perspective configuration, sometimes foreshortening is not desirable, especially when some form of distance measurement needs to be performed in the projected image. By setting the center of projection to infinity, light rays coming from the scene become parallel, and the view volume from Figure 2.7 becomes a rectangular parallelepiped. This projection configuration is called parallel or orthographic, and it is useful when it is required for equal 3D distances to appear as equal 2D distances on the view plane [25]. Figure 2.8 depicts the differences in the projection formation, considering perspective and orthographic configurations.



(a) Perspective Projection.

(b) Orhtographic Projection.

Figure 2.8: The formation of 2D images considering different positions of the center of projection.

Interactive changes in some of the camera parameters (eye, center, up, and field-of-view) trigger monocular depth perception cues, giving the user an immersive experience.

2.2.2 Color Spaces

To represent color, it is essential to define the color space in which the information is stored. Color spaces are multidimensional spaces in which each dimension represents the different components of color, or channels [41]. Different color spaces are designed either to reflect the characteristics of human vision or to suit specific applications.

The red, green, and blue (RGB) space is based on the tri-chromatic nature of the HVS. Its primary colors were standardized by the Commission Internationale d'Éclairage (CIE) using wavelengths: 700.0nm for red, 546.1nm for green, and 435.8nm for blue [42]. RGB is the most common choice for hardware-oriented applications such as monitors and cameras, stemming from the three-electron-gun arrangement in cathode ray tube (CRT) displays, a legacy still present in modern RGB subpixel displays and sensors [24]. It is an additive system, which means that every color is represented as a mix of its primary colors in different amounts. Component values are typically stored in the range $[0, 1]$ for floating-point formats, or $[0, 255]$ when 8-bit integers [25]. Black corresponds to $(0, 0, 0)$ (absence of light), white to $(255, 255, 255)$ (full intensity of all primaries), and equal values across channels produce shades of gray (e.g., $(127, 127, 127)$).

Because the HVS is more sensitive to luminance (i.e., perceived brightness) than to color, texture information can be represented more efficiently by separating brightness from color components. Color spaces that follow this approach are referred to as chrominance-based: they represent color using chrominance (color-difference) components, where each component captures the difference between a specific RGB channel and the overall brightness. These components are typically designed to be approximately orthogonal and statistically independent.

In YCbCr, the Y channel represents *luma*, an approximation of luminance computed from gamma-corrected RGB values.[1] This is conceptually similar to the black-and-white television signal. The two chrominance channels, Cb (chrominance blue) and Cr (chrominance red), carry the color information. Because the HVS is less sensitive to fine chroma details, these channels can be subsampled or more heavily quantized, making YCbCr particularly suitable for image and video compression [42], [43].

In this work, whenever YCbCr is mentioned, it refers to the digital representation defined in the ITU-R BT.709 standard [44]. The conversion from RGB to YCbCr in BT.709 is given by:

$$\begin{bmatrix} Y \\ Cb \\ Cr \end{bmatrix} = \begin{bmatrix} 0.2126 & 0.7152 & 0.0722 \\ -0.1146 & -0.3854 & 0.5000 \\ 0.5000 & -0.4542 & -0.0458 \end{bmatrix} \begin{bmatrix} R \\ G \\ B \end{bmatrix} + \begin{bmatrix} 0 \\ 0.5 \\ 0.5. \end{bmatrix} \tag{2.4}$$

[1] In the literature, the luma is often denoted Y' to distinguish it from the true luminance Y. Since this work deals exclusively with luma, we simplify the notation and use Y throughout.

2.3 RATE-DISTORTION

Compression algorithms can broadly be divided into lossless and lossy. In lossless compression, the original data is perfectly reconstructed after decoding, with the only goal being to remove statistical redundancy in the signal representation. Because the quality of the reconstructed data is fixed (identical to the source), comparing lossless methods reduces to evaluating their compression efficiency, i.e., the number of bits used. Such techniques are essential for domains where every detail must be preserved, such as text, medical imagery, or scientific data.

In contrast, lossy compression deliberately discards some information deemed perceptually or semantically less relevant, in addition to removing redundancy. By exploiting perceptual irrelevancies, lossy schemes typically achieve much higher compression ratios than their lossless counterparts, which is why they dominate in applications such as photography, streaming video, and 3D media. However, lossy methods must balance two competing objectives: minimizing bitrate R and minimizing distortion D. In other words, the goal is to achieve the lowest possible distortion for a given bitrate, or equivalently, the lowest bitrate for a given distortion level.

In practice, distortion in lossy compression is mainly controlled through quantization, which determines how many bits are used to represent each coding unit (e.g., transform coefficient, block, or parameter). Thus, the allocation of bits across different units directly influences both the total bitrate and the resulting quality. We can formalize this by letting $\mathbf{B} = \{b_1, b_2, \ldots, b_m\}$ denote a bit allocation vector, where b_m is the number of bits assigned to the m-th coding unit. Each allocation $\mathbf{B}$ corresponds to a specific operating point on the rate-distortion (RD) plane, as illustrated in Figure 2.9.

The constrained rate allocation problem can be formulated as

$$\min_{\mathbf{B}} D(\mathbf{B}), \quad \text{subject to} \quad R(\mathbf{B}) \leq R_{max}. \tag{2.5}$$

In principle, one could solve (2.5) by exhaustively testing all possible allocations, guaranteeing optimality. However, the search space grows exponentially with the number of coding units m, making this approach computationally infeasible in practice. Simpler heuristics such as greedy allocation, where bits are iteratively assigned to the unit yielding the largest distortion reduction per added bit, are computationally tractable but generally suboptimal, since locally optimal decisions do not necessarily lead to the best global allocation [45].

From information theory [46], we know that the set of achievable RD points has a convex shape, with the optimal operating points lying on the convex hull of this set (Figure 2.9). Directly solving the constrained problem in (2.5) is often intractable. To address this, Shoham and Gersho [47] proposed a relaxed formulation using a Lagrangian cost function

$$J = D(\mathbf{B}) + \lambda R(\mathbf{B}), \quad \lambda \geq 0, \tag{2.6}$$

where λ is a multiplier that controls the trade-off between rate and distortion.

They showed that for any fixed λ, the minimizer $\mathbf{B}^*(\lambda)$ of (2.6) is also the solution to the constrained problem of minimizing $D(\mathbf{B})$ subject to $R(\mathbf{B}) \leq R(\mathbf{B}^*(\lambda))$. In other words, minimizing

the unconstrained Lagrangian cost function produces the same optimal allocation as solving the constrained problem for the specific rate $R(\mathbf{B}^*(\lambda))$. Of course, for an arbitrary target rate R_{max}, there is no guarantee that $R(\mathbf{B}^*(\lambda)) = R_{max}$. However, if we can establish a relationship between λ, the achieved rate, and the resulting distortion, then an iterative search over λ allows us to approach the desired constraint closely.

If we define

$$D(\mathbf{B}) = \sum_i D_i(R_i) \quad \text{and} \quad R(\mathbf{B}) = \sum_i R_i, \tag{2.7}$$

where $D_i(R_i)$ is the distortion of coding unit i when allocated R_i bits, then the Lagrangian cost can be expressed as

$$J = \sum_i D_i(R_i) + \lambda \sum_i R_i. \tag{2.8}$$

Because the sum is separable, minimizing J over all i reduces to minimizing each term individually. In other words, the global optimum can be obtained by solving N local problems, one per coding unit, and combining the results. To find the minimum of J for the i-th coding unit, we set its derivative with respect to R_i to zero:

$$\frac{\partial J}{\partial R_i} = \frac{\partial D_i(R_i)}{\partial R_i} + \lambda = 0, \tag{2.9}$$

thus,

$$\frac{\partial D_i(R_i)}{\partial R_i} = -\lambda, \quad i = 1, 2, \ldots, N. \tag{2.10}$$

Thus, the optimal bit allocation corresponds to the point where each coding unit achieves the same marginal reduction in distortion per additional bit. Which means that all operating points on the convex hull of the RD curve share the same slope, given by $-\lambda$. Intuitively, λ represents the RD trade-off: when λ is small, the optimization favors distortion reduction (high rate); when λ is large, it favors rate savings (higher distortion). Figure 2.9(a) illustrates this concept.

This property is highly valuable, since it reduces the complexity of the global problem to a set of local decisions. In image and video coding, for example, rate-distortion optimization (RDO) can be performed independently at the block level while still yielding a globally optimal solution [48], [49]. As long as both distortion and rate remain additive across coding units, the overall optimization can be achieved through this separable structure. In Chapter 5, we will revisit this concept in the context of 3D assets, showing how the same principle applies when optimizing across different components of a scene.

When targeting specific rates, an operational point of a codec may fall below the theoretical convex hull of the RD curve, due to algorithmic constraints and quantization of encoding parameters. In scenarios where fine-grained rate control is required, it is often practical to consider a more relaxed set that includes such suboptimal points. The boundary of this set, where no further improvement in rate or distortion is possible without degrading the other, is known as the Pareto frontier (see Figure 2.9(b)).

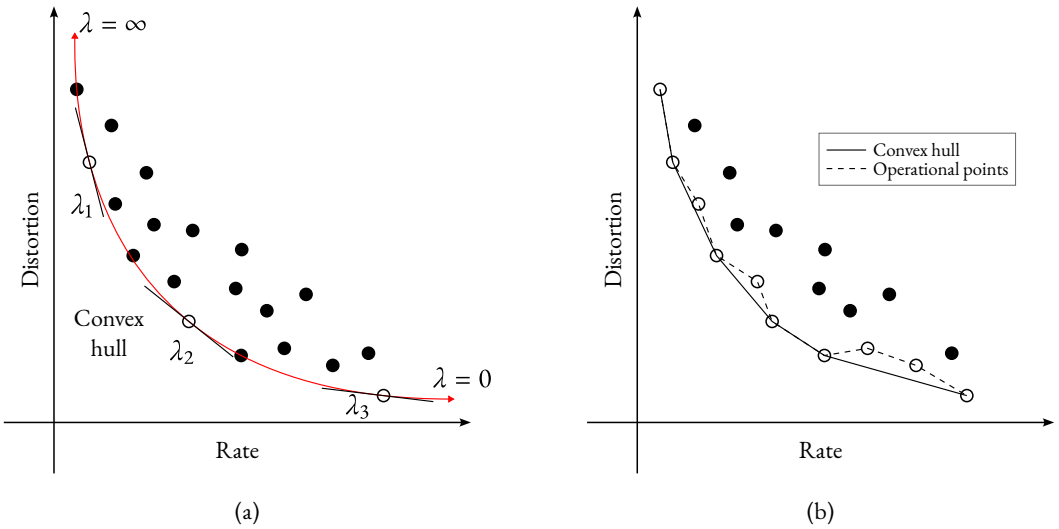


Figure 2.9: Rate-distortion points for different encoder configurations. (a) Optimal operating points lie on the convex hull of the RD curve, each corresponding to a particular trade-off parameter λ. (b) In practice, due to discrete encoding parameters and rate constraints, operational points may fall below the convex hull, e.g., in the Pareto frontier.

Note that, in strict terms, the rate-distortion $R(D)$ and distortion-rate $D(R)$ functions differ only in whether we plot the distortion on the ordinate and the rate on the abscissa, or vice-versa, and they can be used interchangeably depending on convenience. Similarly, the Lagrangian multiplier λ in (2.6) can be expressed multiplying either the rate or the distortion term. The underlying principles of the theory above remain unchanged, with only units and interpretations needing adjustment to match the chosen formulation. It is also common to replace distortion measures (e.g., MSE) with quality measures (e.g., PSNR). Again, this does not change the underlying optimization principles, but it does alter the visual shape of the curves: while distortion typically *decreases* monotonically with rate, quality measures *increase* monotonically. In this thesis, we adopt the term "rate-distortion" (or RD) generically, following common usage in the literature. When relevant, the specific measure employed and whether rate or distortion is treated as the dependent variable will be clarified in context.

2.3.1 Performance Evaluation: Bjøntegaard Delta

RD curves are the standard tool for evaluating and comparing the performance of different codecs. By plotting rate against distortion (or quality), one can assess not only the absolute reconstruction quality at a given bitrate, but also the relative coding efficiency across a range of operating points.

Visual inspection of RD plots in Figure 2.10 gives an intuitive sense of which codec is superior. For instance, when considering distortion (Figure 2.10(a)), the best codec will typically have a curve that lies consistently below the others, indicating that for a similar rate, lower distortion is achieved. However, comparing curves visually can be cumbersome, especially when differences are small, or when RD points are not well-aligned.

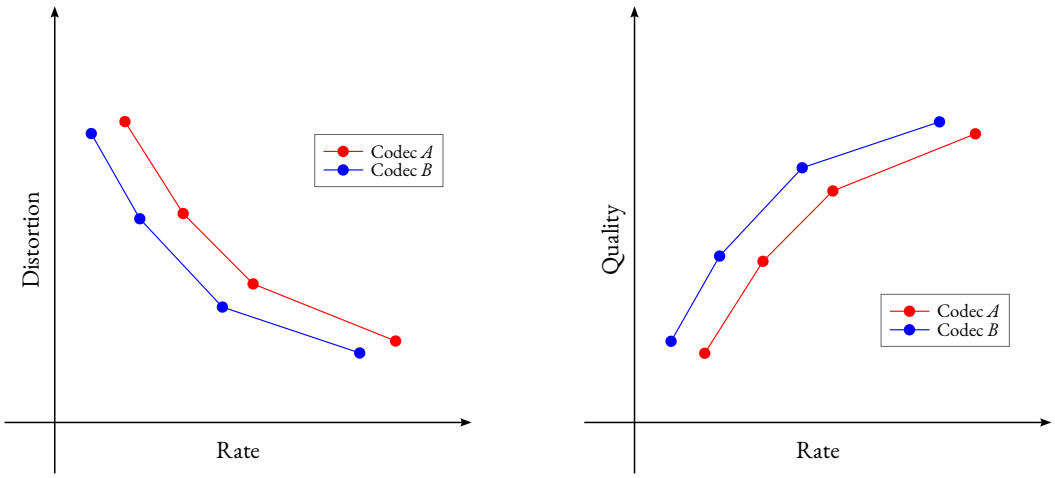


Figure 2.10: Example of operational RD points of two codecs. Codec B is superior to Codec A in this scenario, as it is *below* the RD curve of Codec A when *distortion* is considered (a), and *above* the RD curve of Codec A when *quality* is considered (b).

To simplify performance comparison and express the difference between codecs as a single numerical value, the Bjøntegaard Delta (BD) metric was introduced [50]. The BD metric computes the average difference between two interpolated RD curves, each defined by the discrete operating points of the codecs under comparison.

The BD metric takes as input two sets of RD points, say $\{(x_i, y_i)\}$ and $\{(u_i, v_i)\}$. Each set is first interpolated into continuous functions $y(t)$ and $v(t)$, where t represents a common scale (e.g., rate or distortion). The BD value is then defined as the average area between these two interpolated curves over their overlapping domain

$$\mathrm{BD} = \frac{1}{t_1 - t_0} \int_{t_0}^{t_1} \big[y(t) - v(t) \big] dt, \tag{2.11}$$

where $[t_0, t_1]$ denotes the interval common to both curves. These areas are illustrated in Figure 2.11.

It is important to note that the integration in (2.11) is actually performed in the logarithmic domain, which emphasizes small differences and reduces the impact of large ones compared to a direct linear-domain calculation. As a result, the BD value can be interpreted as the mean relative difference averaged on a logarithmic scale. For a detailed description of the exact interpolation and integration procedures, the reader is referred to [51].

Depending on the axis chosen for integration, the metric has different interpretations.

- **BD-distortion**: integration along the rate axis, which yields the average distortion (or quality) gain (e.g., in PSNR) of one codec relative to another (Figure 2.11(a)). When PSNR is used as the quality metric, the measure is called BD-PSNR and is expressed in decibels (dB). Intuitively, this indicates how much higher the average quality of codec B is compared to codec A at the same bitrates. For example, if codec A is taken as the reference and the computed BD-PSNR is +5 dB, then codec B delivers, on average, 5 dB higher quality at equivalent rates.

- **BD-rate**: integration along the distortion (or quality) axis, which yields the average percentage bitrate savings of one codec relative to another at equivalent quality (Figure 2.11(b)). A negative BD-rate value indicates that the second codec requires fewer bits to reach the same quality. For instance, a BD-rate of −10% (with codec A as reference) means that codec B achieves the same quality as A while using, on average, 10% fewer bits. Conversely, a positive BD-rate implies that the second codec consumes more bits than the first for equivalent quality.

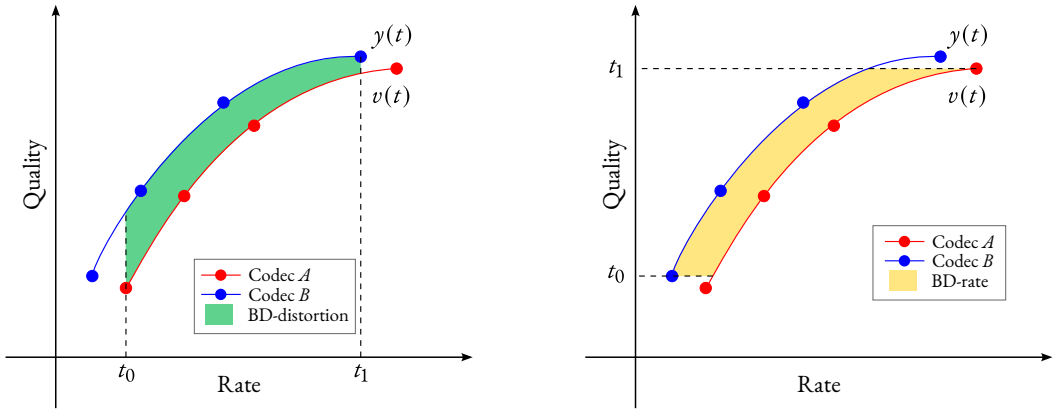


Figure 2.11: Illustration of the area calculation for the BD metric. The green area in (a) represents the integral when computing BD-distortion (along the rate axis), while the yellow area in (b) represents the integral when computing BD-rate (along the distortion axis).

Although practical and widely adopted, the BD metric has several limitations that must be considered when interpreting results. First, if there is no overlapping interval between two RD curves or if the interval is too small, the BD metric cannot be computed. Moreover, the BD metric relies on interpolation of discrete RD points, which may introduce inaccuracies, particularly when the underlying RD curves are not smooth or well-behaved. Its outcome is sensitive to both the interpolation method and the specific RD points chosen for comparison. As a result, different interpolation techniques or point selections can produce different BD values, making cross-study comparisons difficult and potentially misleading [51].

Another important issue is that the BD metric can lead to contradictory conclusions in certain cases. Consider the crossing RD curves in Figure 2.12: (a), BD-PSNR may be negative if region A_2 is larger than A_1, whereas in (b), BD-rate may also be negative if A_3 exceeds A_4. These two results suggest opposite conclusions about which codec performs better, highlighting the metric's limitations when curves intersect [52].

Finally, the BD metric provides only a single averaged value, which, although useful for decision-making, may fail to capture all relevant aspects of codec performance. For instance, it does not consider encoding/decoding complexity, latency, or robustness to transmission errors, all of which are critical in practical applications [51], [52]. Despite these limitations, the BD metric remains an important tool for codec evaluation and comparison, especially when combined with complementary

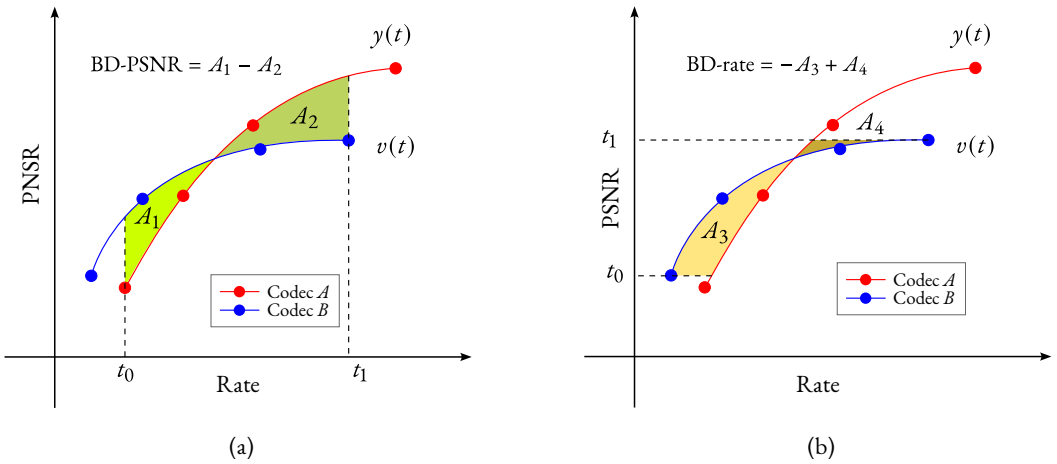


Figure 2.12: Illustration of BD metric limitations for crossing RD curves [52]. Depending on the integration axis, BD-PSNR (a) and BD-rate (b) may lead to contradictory conclusions.

metrics and qualitative assessments.

2.4 3D CONTENT QUALITY ASSESSMENT

When dealing with lossy compression, it is necessary to measure the reconstructed 3D representation's visual fidelity. Quality assessment metrics allow for monitoring and improving the Quality of Experience (QoE) offered to users, which, in turn, guides the design and optimization of compression codecs and processing tools. Quality assessment methods are classified as subjective or objective, depending on whether they are based on human judgment or computed using predefined mathematical criteria. Subjective assessments are appraised for setting the ground truth about the perceived quality of a degraded model since they directly reflect the end-users' opinion. Nevertheless, subjective testing is highly time- and money-consuming. Objective metrics, on the other hand, are designed to provide a quick and cheap prediction of subjective metrics. They usually employ geometric or statistical distance measures between the original and distorted content to approximate the perceived subjective quality.

As discussed in Section 2.2, 3D contents cannot be directly visualized; they must be rendered first, which implies that choices about the rendering algorithm must be made. This makes quality assessment of 3D contents especially challenging. Even the chosen reference method is subject to open questions, such as whether the renderer should preserve the original data as closely as possible or adapt it to better suit the intended application, and whether lighting effects should be applied or not. Objective metrics are also challenging and sometimes controversial, since most of the available metrics are not highly correlated with subjective scores [53]. Moreover, this correlation may vary depending on how the subjective assessment was made [54].

In this work, we employ point-based objective metrics, which are the standard in MPEG 3DGH

evaluations and are available through the `dmetric`[2] software [55]. During the assessment of the V-DMC standard, it was observed that point-based metrics alone were insufficient to accurately capture the perceptual quality of reconstructed meshes. To address this limitation, a new metric suite, `mmetric`[3], was developed, incorporating projection-based measures as well [56]. In addition to these MPEG-standardized metrics, we also consider previously proposed projection-based approaches [57].

2.4.1 Point-based Metrics

The objective metrics used in MPEG 3DGH evaluation procedure are full-reference quality metrics, which means that the original point cloud is used in its entirety in the comparison. They are computed symmetrically, first considering the original point cloud $\mathcal{O}$ as the reference, and then considering the degraded version $\mathcal{D}$ as the reference. The metrics are computed by measuring the distance between each point in $\mathcal{D}$ and its nearest neighbor in $\mathcal{O}$, quantifying the individual degradation at each point and averaging these values to obtain an overall quality score.

Each metric is named according to how the error between the reconstructed and reference points is computed. Errors measured directly between corresponding points define the point-to-point metric; projecting the error along the reference's normal direction yields the point-to-plane metric; and comparing color differences between corresponding points results in the end-to-end color metric. Typically, the Euclidean distance is used, although the Hausdorff distance is also available. The Hausdorff distance measures how far two subsets of a metric space are from each other. In other words, it represents the greatest distance from any point in one set to its closest point in the other [58].

For all point-based metrics, it is necessary to find, for each query voxel $\mathbf{v}_{\mathcal{O}}(k)$ in $V_{\mathcal{O}}$ of the original point cloud $\mathcal{O}$, the corresponding nearest neighbor $\mathbf{v}_{\mathcal{D}}(k_{\mathcal{O}})$ in $V_{\mathcal{D}}$ of the degraded point cloud $\mathcal{D}$,

$$\mathbf{v}_{\mathcal{D}}(k_{\mathcal{O}}) = \underset{\forall \mathbf{v}_{\mathcal{O}}(k) \in V_{\mathcal{O}}}{\arg\min} \|\mathbf{v}_{\mathcal{O}}(k) - \mathbf{v}_{\mathcal{D}}(i)\|, \tag{2.12}$$

and symmetrically,

$$\mathbf{v}_{\mathcal{O}}(k_{\mathcal{D}}) = \underset{\forall \mathbf{v}_{\mathcal{D}}(k) \in V_{\mathcal{D}}}{\arg\min} \|\mathbf{v}_{\mathcal{D}}(k) - \mathbf{v}_{\mathcal{O}}(i)\|, \tag{2.13}$$

which is performed using a nearest neighbor search algorithm. The principles of point-based metric computation are illustrated in Figure 2.13.

Since no attribute is evaluated in the point-to-point metric, neighboring points within the same distance are discarded, and only the first found neighbor is used. However, when considering attributes, as is the case of point-to-plane and point-based color metrics, neighboring points within the same distance need to be included, as their attribute values may differ.

[2] https://github.com/MPEGGroup/mpeg-pcc-dmetric
[3] https://github.com/MPEGGroup/mpeg-pcc-mmetric

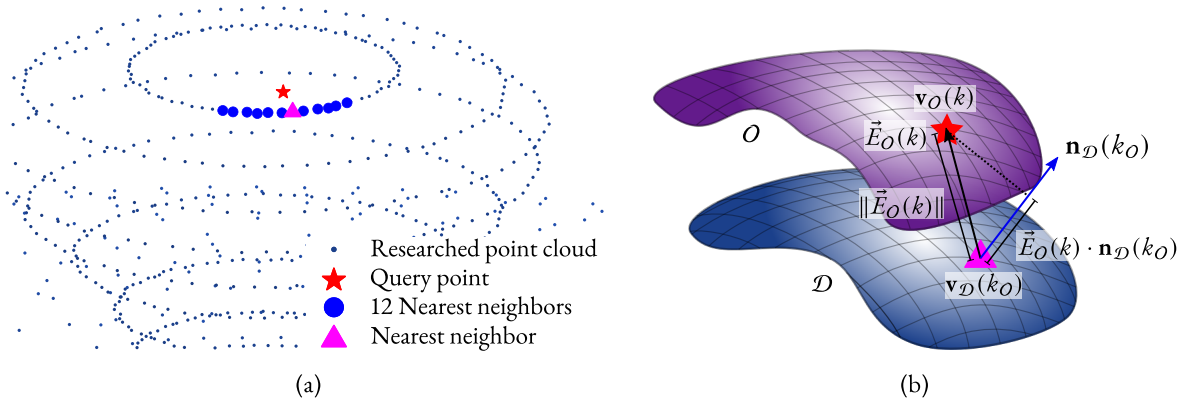


Figure 2.13: Point-based metric illustration. In (a), the nearest neighbor search was performed to find the 12 nearest neighbors of the query point depicted with a red star in the researched point cloud. The first nearest neighbor is identified with a magenta triangle. In (b), the point-to-point and the point-to-plane errors between the query point and its nearest neighbor are graphically represented.

2.4.1.1 Point-to-point metric (D1)

Point-to-point metrics were first introduced in order to measure data acquired by LiDAR scanners [59]. It is a simple and fast method, and it does not require any additional information but both point clouds geometries. In the `dmetric` software, it is referred to as the D1 metric.

Using $\mathcal{O}$ as reference, the error vectors $\vec{E}_{\mathcal{O}}(k)$ are computed between every point in $V_{\mathcal{O}}$ and its corresponding nearest neighbor in $V_{\mathcal{D}}$, which were found using (2.12),

$$\vec{E}_{\mathcal{O}}(k) = \mathbf{v}_{\mathcal{O}}(k) - \mathbf{v}_{\mathcal{D}}(k_{\mathcal{O}}). \tag{2.14}$$

To compute the mean-squared error (MSE), the squared norms of all error vectors are averaged over the $K_{\mathcal{O}}$ points,

$$\mathrm{D1}_{\mathcal{O}}^{\mathrm{MSE}} = \frac{1}{K_{\mathcal{O}}} \sum_{k=1}^{K_{\mathcal{O}}} \|\vec{E}_{\mathcal{O}}(k)\|^2. \tag{2.15}$$

Figure 2.13(b), illustrates the graphic representation of the error of (2.14).

The error calculated by (2.15) estimates the degradation of $\mathcal{D}$ using just the subset of voxels $\{\mathbf{v}_{\mathcal{D}}(k_{\mathcal{O}})\}$, the nearest neighbors of $\mathcal{O}$ in $\mathcal{D}$. When $\mathrm{D1}_{\mathcal{O}}^{\mathrm{MSE}} = 0$, it means that $\mathcal{D}$ contains all the points of $\mathcal{O}$, in other words $V_{\mathcal{D}} \supset V_{\mathcal{O}}$. Notice, however, that Equation (2.15) does not necessarily consider all the elements in $V_{\mathcal{D}}$. To account for all points in both clouds, a symmetrical error is also computed by taking the degraded point cloud $\mathcal{D}$ as the reference,

$$\vec{E}_{\mathcal{D}}(k) = \mathbf{v}_{\mathcal{D}}(k) - \mathbf{v}_{\mathcal{O}}(k_{\mathcal{D}}), \tag{2.16}$$

$$\mathrm{D1}_{\mathcal{D}}^{\mathrm{MSE}} = \frac{1}{K_{\mathcal{D}}} \sum_{k=1}^{K_{\mathcal{D}}} \|\vec{E}_{\mathcal{D}}(k)\|^2. \tag{2.17}$$

In turn, when $\mathrm{D1}_{\mathcal{D}}^{\mathrm{MSE}} = 0$, it means that all the points of $\mathcal{D}$ are contained in $\mathcal{O}$, or $V_{\mathcal{D}} \subset V_{\mathcal{O}}$. Hence, (2.15) and (2.17) are complementary. In order to arrive at the final MSE measure, either the

maximum or the average of the two errors must be calculated [59]–[61]. At the current version of the `dmetric` software, the maximum error is used, such that,

$$\mathrm{D1}^{\mathrm{MSE}} = \max\left(\mathrm{D1}_{\mathcal{O}}^{\mathrm{MSE}},\ \mathrm{D1}_{\mathcal{D}}^{\mathrm{MSE}}\right). \tag{2.18}$$

2.4.1.2 Point-to-plane metric (D2)

Although the D1 metric captures the differences between the points, it fails to account that the points in a point cloud represent a surface. For that matter, the point-to-plane metric was included in `dmetric` and is referred to as the D2 metric. Using $\mathcal{O}$ as reference, the metric is calculated by projecting, i.e., taking the dot product, of each error vector $\vec{E}_{\mathcal{O}}(k)$ along the normal direction of its nearest neighbor in $\mathcal{D}$, $\mathbf{n}_{\mathcal{D}}(k_{\mathcal{O}})$, as illustrated in Figure 2.13(b). Symmetrically, when $\mathcal{D}$ is the reference, each $\vec{E}_{\mathcal{D}}(k)$ is projected along the normal direction of its nearest neighbor in $\mathcal{O}$, $\mathbf{n}_{\mathcal{O}}(k_{\mathcal{D}})$. This way, larger penalties are imposed on errors that move further away from the local plane surface [61], which creates a perceived rugged surface; and smaller penalties are imposed for errors that move perpendicular to the local plane, which does not change the perceived surface smoothness.

The D2 metric is evaluated as follows:

$$\mathrm{D2}^{\mathrm{MSE}} = \max\left\{\frac{1}{K_{\mathcal{O}}}\sum_{k=1}^{K_{\mathcal{O}}}\left(\vec{E}_{\mathcal{O}}(k)\cdot\mathbf{n}_{\mathcal{D}}(k_{\mathcal{O}})\right)^2,\ \frac{1}{K_{\mathcal{D}}}\sum_{k=1}^{K_{\mathcal{D}}}\left(\vec{E}_{\mathcal{D}}(k)\cdot\mathbf{n}_{\mathcal{O}}(k_{\mathcal{D}})\right)^2\right\}. \tag{2.19}$$

When normal vectors $N_{\mathcal{O}} = \{\mathbf{n}_{\mathcal{O}}(k)\}$ are not available in $\mathcal{O}$, they need to be estimated for the D2 calculations. That is done by assuming that the point cloud surface can be locally modeled by a plane [62]. Estimation of the normal vectors $N_{\mathcal{D}} = \{\mathbf{n}_{\mathcal{D}}(k)\}$ in $\mathcal{D}$ is not carried out the same way, however, since they are likely to be biased due to geometric distortions. Instead, normals from $\mathcal{O}$ are transferred to the voxels in $\mathcal{D}$ using their nearest neighbors [61].

The peak signal-to-noise ratios (PSNR) in dB for both D1 and D2 metrics is given by:

$$\mathrm{PSNR}_{\mathrm{D1}} = 10\log_{10}\left(\frac{p_g^2}{\mathrm{D1}^{\mathrm{MSE}}}\right), \tag{2.20}$$

$$\mathrm{PSNR}_{\mathrm{D2}} = 10\log_{10}\left(\frac{p_g^2}{\mathrm{D2}^{\mathrm{MSE}}}\right). \tag{2.21}$$

For the point clouds defined in the PCC CTC [63] dataset, the peak value p_g is a constant defined in Table 2. As a rule of thumb, p_g is the length of the bounding cube's diagonal containing the point cloud for contents in Categories 1 and 2. For Category 3, the intrinsic resolution is used, which in the `dmetric` is defined as the maximum distance between neighboring voxels in the original point cloud.

2.4.1.3 End-to-end color metrics

After finding a point-to-point correspondence using (2.13) and (2.12), texture (or color) metrics for point clouds are calculated the same way MSE is performed in images. Colors are first converted to YCbCr space following the BT.709 standard [44], then the MSE is calculated separately for each channel $\vartheta \in \{\mathrm{Y}, \mathrm{Cb}, \mathrm{Cr}\}$,

$$\mathrm{MSE}_{\vartheta} = \max\left\{ \frac{1}{K_O} \sum_{k=1}^{K_O} \left(\vartheta_O(k) - \vartheta_D(k_O)\right)^2, \frac{1}{K_D} \sum_{k=1}^{K_D} \left(\vartheta_D(k) - \vartheta_O(k_D)\right)^2 \right\}. \tag{2.22}$$

It is worth noting that neighboring points within the same distance have their colors averaged for the MSE computation.

For each color component $\mathrm{PSNR}_{\vartheta}$ is calculated by

$$\mathrm{PSNR}_{\vartheta} = 10 \log_{10}\left(\frac{p_c^2}{\mathrm{MSE}_{\vartheta}}\right), \tag{2.23}$$

with $\vartheta \in \{\mathrm{Y}, \mathrm{Cb}, \mathrm{Cr}\}$, and $p_c = 1$ when considering a floating-point representation, or $p_c = 255$ when considering an 8-bit color depth integer representation.

2.4.2 Projection-based Metrics

As considered in the volume visualization pipeline of Figure 2.6, 3D contents are in fact consumed as 2D projections, and therefore it makes sense to evaluate its quality using 2D projections. This approach was first used in [57], where the rendered point clouds were mapped onto planar surfaces, then 2D image quality assessment metrics were used to evaluate compression performance. More recently in [56], an image-based sampling metric (IBSM) was proposed, which is similar to the previous work, but presents a more systematic approach for getting the projections. Projection-based metrics are interesting because they provide a quality metric that considers the 3D content as a whole, differently from the point-based metrics that consider geometry and attributes separately. Also, for applications with a well-defined rendering methodology and well-known use cases (i.e., which viewpoints are more important, the most used camera settings for visualization, etc.), projection-based metrics can be fine-tuned to mimic those specific conditions to better approximate the human subjective perception of the contents.

A simple approach to get the projections is to consider the six axis-aligned orthographic projections around the bounding box of the point cloud or mesh. This way, it is possible to have a one-to-one correspondence between voxels and pixels, for the case of point clouds. As a result, the camera parameters are set to acquire projections that are perfectly aligned with the bounding cube containing the 3D content. This ensures that a point cloud of bit-depth 10, for example, will have six bitmap images of 1024-by-1024 pixels, as illustrated by Figure 2.14. Since both point clouds and meshes typically represent individual objects within a larger space, most of the bounding volume remains unoccupied, making their representation inherently sparse. Hence, most of the pixels of the

projected bitmaps do not belong to the effective part of the content (i.e., background color), so it is important to choose a value of background color to least interfere with the calculations. Moreover, it has been found that excluding part of the background pixels improves the accuracy of the metric prediction [64]. This way, image quality metrics are applied to the six pairs of projections, and the total score is computed as the average of the metrics. In this work, three different 2D metrics were selected: the MSE/PSNR, which is based on pixels differences and relates with the point-based metrics of the last Section; Structural Similarity Index Measure (SSIM), which searches for similarities in patches rather then pixel-wise; and the Visual Information Fidelity (VIF) measure, which is an information theory approach to model how image information is perceived in the HVS [65].

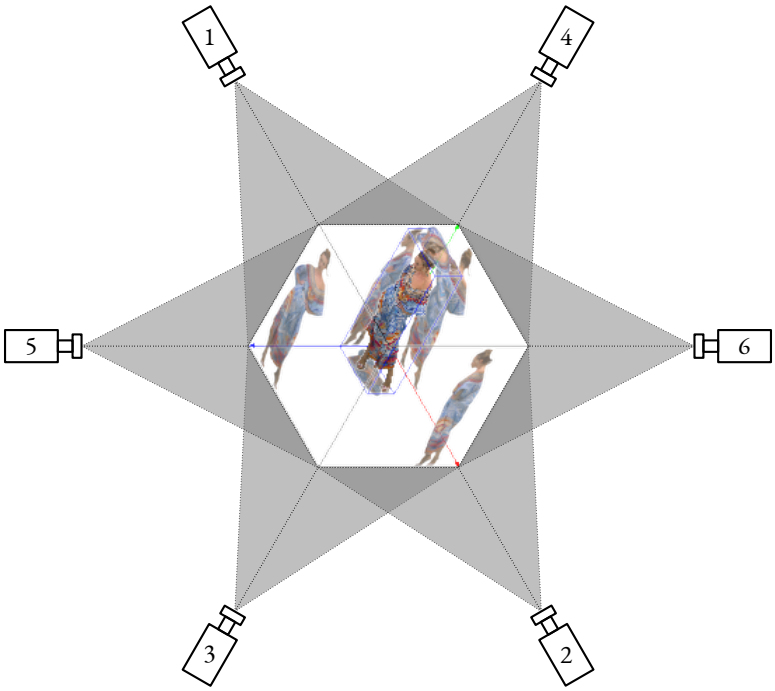


Figure 2.14: The six camera positions used to get axis-aligned projections.

2.4.2.1 Projected MSE and PSNR

MSE is used for measuring the differences of pixels between a reference image and its distorted version. More specifically, its scale-agnostic version, the PSNR, is widely used for evaluating the degradation of compression artifacts of images. The MSE computation of an RGB image is analogous to the one presented in Section 2.4.1. The error vector's squared norm in the RGB space between each pixel in the reference image and its counterpart in the distorted image is computed. Thus, to calculate the projected MSE (PMSE) of a point cloud, the MSEs of the six pairs of W_{n_v} by H_{n_v} images are averaged:

$$\text{PMSE} = \frac{1}{6}\sum_{n_v=1}^{6}\left(\frac{1}{W_{n_v}H_{n_v}}\sum_{x=1}^{W_{n_v}}\sum_{y=1}^{H_{n_v}}\|\text{RGB}_{\text{ref}}^{n_v}(x,y) - \text{RGB}_{\text{dist}}^{n_v}(x,y)\|^2\right). \tag{2.24}$$

The projected PSNR (PPSNR), for 8-bit color depth images, is defined as,

$$\text{PPSNR} = 10 \log_{10} \left(\frac{255^2}{\text{PMSE}} \right). \tag{2.25}$$

2.4.2.2 Projected SSIM

The SSIM assesses perceptual quality between reference and distorted images by comparing local similarities on spatial patches, assuming that the HVS is highly adapted to extract structural information from visual scenes. It was created from the perspective of image formation [66], so it performs three comparisons (luminance, contrast, and structure) on each patch, and then averages the comparisons over the whole image to arrive at an overall image quality, which is referred to as the mean SSIM (MSSIM). As the SSIM was developed for grayscale images, we use it on the luma channel Y only. Following the implementation provided by [67], in its default configuration, the SSIM between signals x and y is computed using 11×11 Gaussian windows with $\sigma = 1.5$ in each image:

$$\text{SSIM}(x, y) = \frac{(2\mu_x\mu_y + C_1)(2\sigma_{xy} + C_2)}{(\mu_x^2 + \mu_y^2 + C_1)(\sigma_x^2 + \sigma_y^2 + C_2)}, \tag{2.26}$$

where μ represents the mean, σ^2 the variance, $C_1 = (0.01 \cdot 255)^2$, and $C_2 = (0.03 \cdot 255)^2$. The MSSIM is, then, the average of the M patches over the entire images X and Y,

$$\text{MSSIM}(X, Y) = \frac{1}{M} \sum_{j=1}^{M} \text{SSIM}(x_j, y_j). \tag{2.27}$$

Finally, the projected SSIM (PSSIM) is calculated as the average of the MSSIM for every viewpoint projection pair,

$$\text{PSSIM} = \frac{1}{6} \sum_{n_v=1}^{6} \text{MSSIM}(X_{n_v}, Y_{n_v}). \tag{2.28}$$

2.4.2.3 Projected VIF

The information theory approach of VIF [68] measures the fidelity of a distorted image by calculating the loss of image information to the distortion process and comparing it with its reference counterpart. Thus, exploring the relationship between image information and visual quality. Figure 2.15 depicts a block diagram of the information content model used in VIF.

The reference image is assumed to be the output of a stochastic "natural" source and is modeled as a random field C using a Gaussian Scale Mixture (GSM). After passing through the HVS channel, it becomes $\mathcal{E}$, representing the information processed by the brain. The information content of the reference image is quantified as being the mutual information between C and $\mathcal{E}$, $I(C; \mathcal{E})$. Ideally, this would represent the information that the brain could extract from the output of the HVS. The information content of the distorted image is likewise quantified, but a distortion channel $\mathcal{D}$ is introduced before the HVS, such that it becomes the mutual information between C and

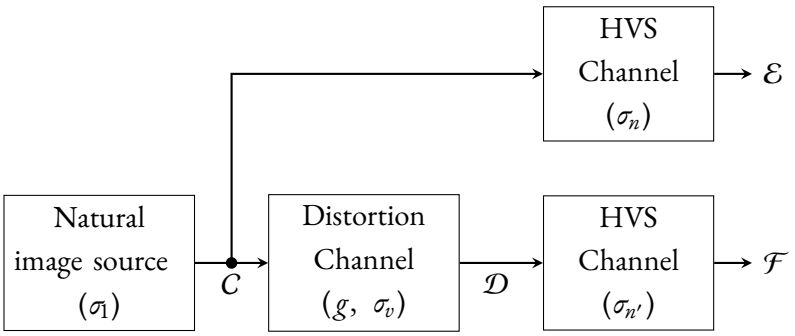


Figure 2.15: Information content model used in VIF [65].

$\mathcal{F}$, $I(\mathcal{C};\mathcal{F})$. The VIF measure is, then, calculated as the ratio of the sum of the distorted image information content over the sum of the reference image reference content, across all subbands of the GSM [65], [69],

$$\text{VIF} = \frac{\sum_{j\in \text{ subbands}} I(\mathcal{C}^j;\mathcal{F}^j|s^j)}{\sum_{j\in \text{ subbands}} I(\mathcal{C}^j;\mathcal{E}^j|s^j)}. \tag{2.29}$$

In this work, we use a computationally simpler multi-scale pixel implementation of VIF, the pixel domain VIF (VIFp) [67]. Similar to the SSIM, the VIF was designed for grayscale images only, so again it is applied only for the luma channel. The mutual information is calculated across four subbands. For each subband j, M_j Gaussian windows are considered, such that

$$\text{VIFp} = \frac{\sum_{j=1}^{4}\sum_{i=1}^{M_j}\log_{10}\frac{1+g(i,j)^2\sigma_1(i,j)^2}{\sigma_v(i,j)^2+\sigma_n(i,j)^2}}{\sum_{j=1}^{4}\sum_{i=}^{M_j}\log_{10}\frac{1+\sigma_1(i,j)^2}{\sigma_n(i,j)^2}}. \tag{2.30}$$

The projected VIFp (PVIFp) is, then, the average of the VIFp calculated for each projection pair,

$$\text{PVIFp} = \frac{1}{6}\sum_{n_v=1}^{6}\text{VIFp}_{n_v}. \tag{2.31}$$

2.4.2.4 Image-based Sampling Metric (IBSM)

The computation of the IBSM [56] also relies on computing MSE/PSNR over projected images, as described in Section 2.4.2.1. An overview of the IBSM is shown in Figure 2.16. Sixteen orthographic projections are generated using a Fibonacci sphere lattice (Figure 2.17), with each view direction vd_i pointing toward the center of the bounding sphere enclosing the mesh. Rendering is performed via mesh rasterization with clockwise back-face culling, which omits faces oriented away from the camera to reduce processing and avoid drawing hidden geometry. For each view, the rendering generates:

- a color image: RGB values of the nearest projected triangle; for textured meshes, colors are obtained by bilinear interpolation in texture space, and for per-vertex color meshes, by barycentric interpolation;

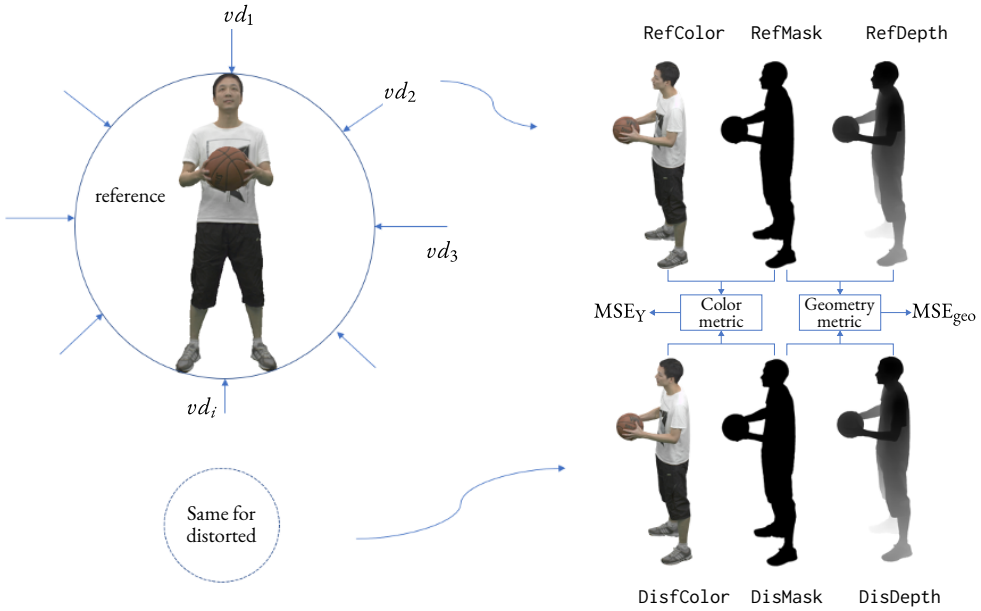


Figure 2.16: Overview of the image-based sampling metric (IBSM) [56]. Ref/DisMask are binary maps where pixel$[i, j] = 1$ if a projection exists in the associated color image, and 0 otherwise. Ref/DisDepth are the corresponding depth images. All images have the same dimensions of 2048×2048 pixels.

- a mask image: binary map indicating whether a projection exists for each pixel; and,
- a depth image: distance from the projection plane to the 3D surface in 3D space.

Color and depth images are then used to compute the luma and geometry metrics, respectively.

Figure 2.17: Example of the IBSM orthographic projections [56].

Let mask be a matrix of same size of maskRef and maskDis,

$$\texttt{mask}(i, j) = \begin{cases} 1, & \text{if} \quad \texttt{maskRef}(i, j) + \texttt{maskDis}(i, j) = 2, \\ 0, & \text{otherwise.} \end{cases} \tag{2.32}$$

Then, let n be the width of the projected images, the combined number of projected pixels `nb_samples` of all the view directions for one frame is:

$$\texttt{nb\_samples} = \sum_{v=1}^{16} \sum_{i=1}^{n} \sum_{j=1}^{n} \texttt{mask}(i, j). \tag{2.33}$$

For the luma metric, a conversion from RGB space to YUV space is performed by using ITU-R BT.709 [44], then,

$$\text{MSE}_{\text{Y}} = \frac{1}{\texttt{nb\_samples}} \sum_{i=1}^{n} \sum_{j=1}^{n} \texttt{mask}(i, j) \cdot \left[Y_{\texttt{ref}}(i, j) - Y_{\texttt{dis}}(i, j) \right]^2. \tag{2.34}$$

For the geometry metric, let $dv(i, j)$ be a sample of the depth image, also let `sig_dynamic` be the dynamic range of the depth signal initialized with the maximum between the diagonals of the bounding boxes of both models, then,

$$\text{MSE}_{\text{geo}} = \frac{1}{\texttt{nb\_samples}} \sum_{i=1}^{n} \sum_{j=1}^{n} \texttt{mask}(i, j) \cdot \left[\frac{dv_{\texttt{ref}}(i, j) - dv_{\texttt{dis}}(i, j)}{\texttt{sig\_dynamic}} \cdot 255 \right]^2. \tag{2.35}$$

This depth renormalization to 255 is used to get geometric MSE and PSNR comparable to the color ones.

The respective PSNR_θ with $\theta \in \{\text{Y}, \text{geo}\}$ are then calculated as:

$$\text{PSNR}_\theta = \min \left[999.99,\ 10 \log_{10} \left(\frac{255^2}{\text{MSE}_\theta} \right) \right]. \tag{2.36}$$

2.5 MPEG 3D GRAPHICS COMPRESSION

MPEG has focused significant efforts on standardizing compression techniques for 3D graphics data, due to the increasing popularity of 3D immersive applications. ISO/IEC JTC1 SC29 WG7 – MPEG 3DGH is the working group responsible for this task,[4] and it has been working on various aspects of 3D graphics compression, including point clouds, meshes, and more recently, 3D Gaussian splatting. Figure 2.18, highlights key milestones in the group's activities.

In 2017, PCC standardization was started with the issue of call for proposals (CfP) [72], which used the work from Mekuria *et al.* [73] as the baseline anchor. The PCC CfP targeted three categories of point clouds: (1) *static objects and scenes* for geographic information systems (GIS) and cultural heritage; (2) *dynamic objects* for XR and tele-immersive applications; and (3) *dynamic acquisition*, such as point clouds captured by LiDAR for autonomous navigation [74].

The development of these standards began with the creation of test models (TM) for each category (TMC) and their respective reference software implementations [75]. In 2018, TMC1 and

[4] Before the reorganization of SC29 inJune 2020, 3DG was a sub-group under WG11 – MPEG [70], [71].

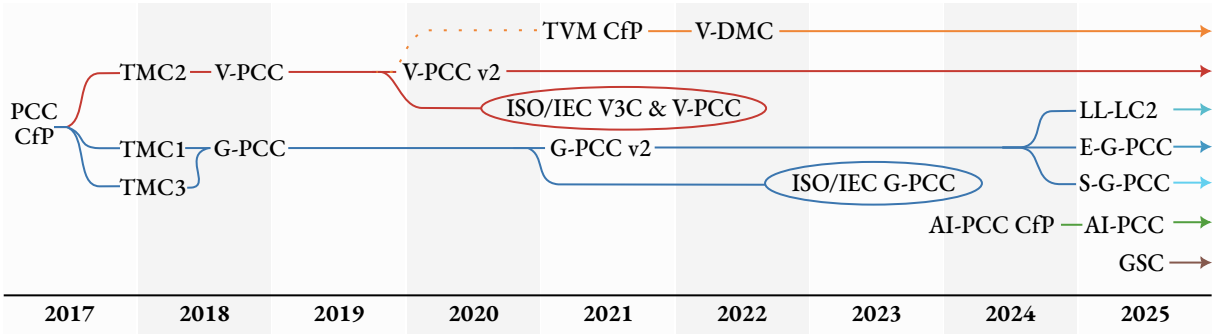


Figure 2.18: MPEG 3DGH standardization timeline.

TMC3 were consolidated into TMC13, which became the basis for the geometry-based point cloud compression (G-PCC) specification. G-PCC was created as a general-purpose solution for compressing point clouds, regardless of their density or temporal characteristics. Meanwhile, TMC2 continued as the reference for video-based point cloud compression (V-PCC) specification. V-PCC relies on the projection of the 3D geometry into 2D patches, building on existing video coding techniques, which allowed for a faster standardization process. Finalized in 2020, V-PCC v1 was published, in the following year, under the broader Visual Volumetric Video-based Coding (V3C) framework and specifications [76], [77], which provides a unified framework for compressing immersive media like point clouds and other volumetric data. G-PCC v1 was completed in 2021, and a second version aiming for advanced compression tools (e.g., inter-frame coding) was started.

That same year, a new CfP [78] was issued, this time for the compression of dynamic time-varying meshes (TVM) within the V3C framework, focusing on video-based approaches. Following the analysis of the proposals [79], in 2022, the Video-based Dynamic Mesh Compression (V-DMC) TM was established, relying on displaced subdivision surface model in combination with video-based coding [80]. In 2024, MPEG 3DGH signalled the integration of artificial intelligence (AI) into PCC workflows, with the launching of the CfP for AI-based PCC (AI-PCC) [81].

At the time of writing, V-PCC v2 remains in development. G-PCC v2 was split into three standards: Low latency, low complexity LiDAR coding (LL-LC2), Enhanced G-PCC (E-G-PCC) and solid G-PCC (S-G-PCC). The V-DMC standard is near finalization, expected to be published by the end of 2025 [11], and AI-PCC is in its working draft (WD) phase. Additionally, with V-PCC, G-PCC and V-DMC fading out, WG7 jointly with WG4 have recently started exploring the compression methods for 3D Gaussian splats [82], [83] (Gaussian splat coding – GSC), a popular representation that models 3D scenes using a sparse set of anisotropic 3D Gaussians, enabling real-time rendering of complex geometry [84].

Next, a brief overview of G-PCC and V-DMC standards are presented, focusing on the main concepts and tools used in the compression process. V-PCC is not covered in this chapter, as it is not the focus of the research. A more detailed explanation of MPEG 3DGH PCC can be found in [10] and in [85].

2.5.1 Geometry-based Point Cloud Compression (G-PCC)

2.5.1.1 Coding Principles

G-PCC is dedicated to generic point clouds, covering both sparse and dense data, as well as static and dynamic objects and scenes. It is designed to handle datasets with complex geometry and irregular point sampling, such as those captured from LiDAR scans, cultural heritage preservation efforts, and other real-world 3D acquisition scenarios. To ensure robustness and adaptability across a wide range of content, G-PCC follows a one-codec-fits-all strategy, integrating various geometry and attribute coding tools to support different use cases.

In its core pipeline, geometry is first processed and represented using efficient data structures (e.g., octrees) that encode occupancy information in a compact binary form. G-PCC's octree-based geometry coding relies on a cascade of binary entropy coders, typically implemented using Context-Adaptive Binary Arithmetic Coding (CABAC) [86], to encode occupancy information [87]. Previously coded occupancy bits serve as context to improve compression efficiency and capture spatial correlations within the tree structure. Attributes, such as color or reflectance, are typically transformed using subband-based decompositions, quantized, and entropy coded also using CABAC. The two resulting bitstreams constitute the compressed point cloud. As G-PCC was designed to handle large-scale point clouds, it includes some support for partial encoding/decoding and enables some parallelization to manage computational demands. Although originally intended for static point clouds, more recent developments in version 2 of the standard have introduced tools for inter-frame coding. A summary of such new tools can be found in [87].

2.5.1.2 Codec Architecture

The software used for testing G-PCC is referred as Test Model Categories 1 and 3 (TMC13) due to the targeted point clouds it was supposed to handle: static and dynamically acquired point clouds (e.g. LiDAR), respectively.[5] The encoding/decoding scheme used in TMC13 is depicted in Figure 2.19. Since G-PCC is agnostic to the input data coordinate representation, there is a coordinate transformation, followed by a voxelization pre-processing step before the geometry coding. Input coordinates are transformed such that all points of the input data lie in a bounding cube $[0, 2^d)^3$, for some non-negative integer parameter d. Voxels are then represented by coordinates representing the center of any of the unit cubes $[i-0.5, i+0.5)\times[j-0.5, j+0.5)\times[k-0.5, k+0.5)$, for i, j, k integers between 0 and $2^d - 1$ [88]. All points from the original set that lie within a voxel's boundary are mapped to that voxel. This mapping step may result in multiple points with the same position—duplicate points. Usually, duplicate points are consolidated into one, and their attributes are averaged.

This pre-processing step conforms data into TMC13 format, allowing for efficient 3D representation. Given that typically only about 1% of the voxels within the bounding cube are occupied, an efficient data structure is necessary to handle sparsity. The octree [89], a hierarchical-tree data

[5] https://github.com/MPEGGroup/mpeg-pcc-tmc13

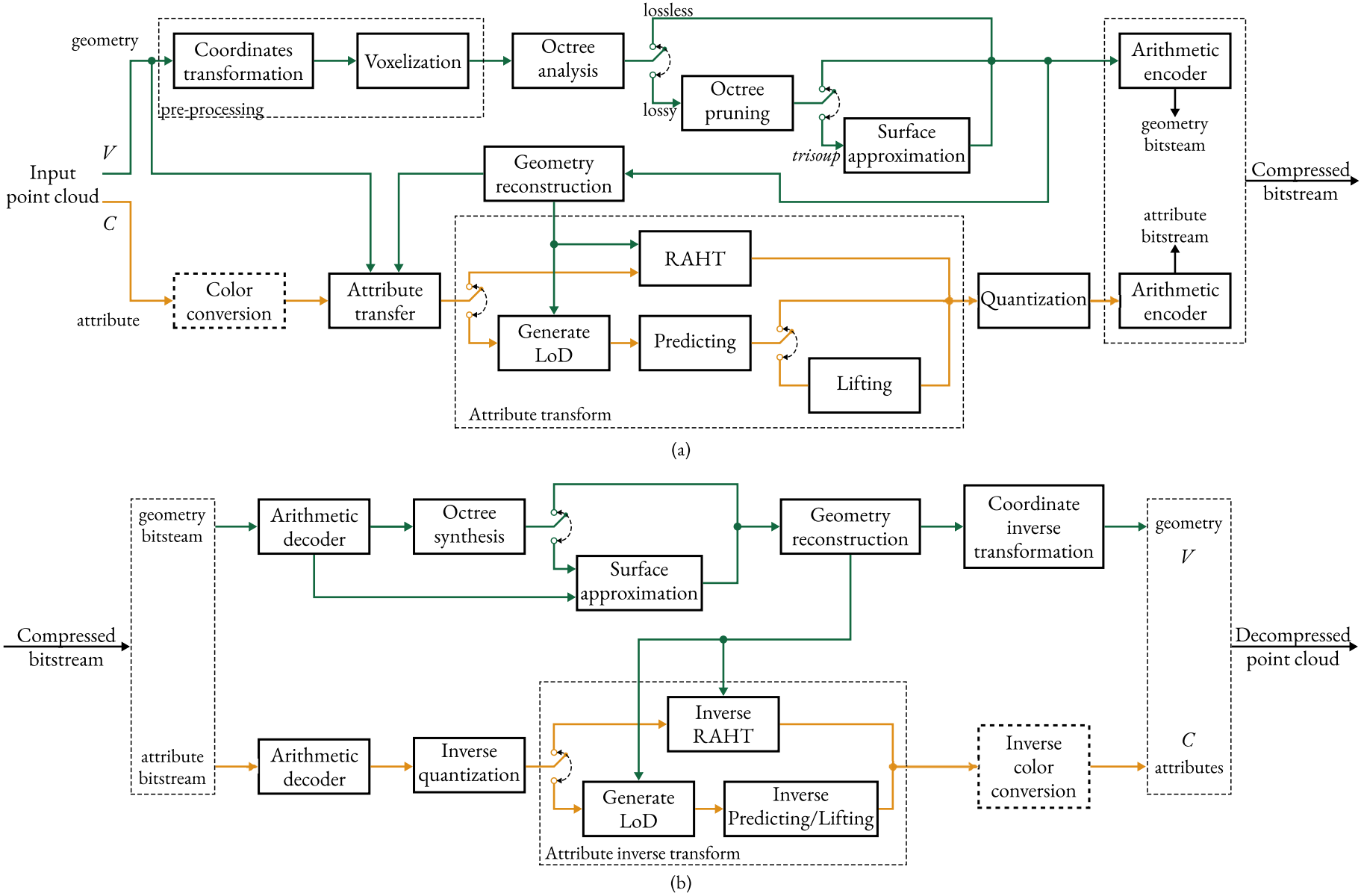
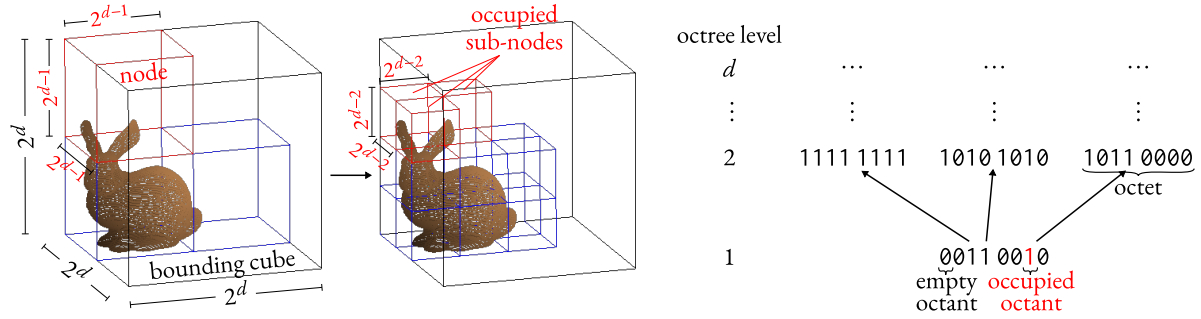


Figure 2.19: TMC13 encoding (a) and decoding (b) schemes [88].

structure that has been shown effective to code sparse 3D objects [90], [91], is the data structure chosen for representing 3D points in TMC13. After the coordinate transformation and voxelization processes, building the octree is straightforward, as illustrated in Figure 2.20.



(a) Graphic representation

(b) Tree representation

Figure 2.20: Octree analysis illustration.

The root node is the $2^d \times 2^d \times 2^d$ bounding cube, which is divided into eight sub-volumes, or octants, with 1/8 of the volume of its parent. Those octants containing points are marked as 1 and further divided; otherwise, they are marked as `0`, and the division stops. At each node division, eight child nodes are generated, represented by a 1-byte word, an octet, in the tree. Thus, each octree level refines the coordinates of the points within a sub-volume by using one bit for each component, at a total cost of eight bits per refinement [88]. The subdivision of occupied nodes is recursively repeated until the octants reach the smallest volume element, a $1 \times 1 \times 1$ cubic voxel. The maximum possible number of subdivision levels is defined by the parameter d, the octree depth, and it is related to the point cloud's sampling resolution.

Occupancy bits in G-PCC are encoded using a variety of context models that exploit the occupancy information of previously coded neighboring nodes. This results in a large number of possible context configurations. To manage this complexity, configuration reduction techniques are employed, such as grouping symmetric or rotated configurations, and an intra-prediction technique [92] is used to reduce the prediction space by classifying neighbors into a ternary state: occupied, unoccupied, or not predicted. In addition, since several binary entropy coders are used, G-PCC introduces a mechanism called Optimal Binarization with Update on-the-Fly (OBUF) [93]. OBUF maps the contextual configuration of each bit to a specific binary coder using a look-up table (LUT), where each entry corresponds to a neighbor configuration. The selected coder index determines which entropy coder will be used, and the LUT is dynamically updated based on a memory channel model that adapts to the evolving coding statistics [87].

For lossless geometry compression, all the octets of the tree are used. In dense areas of the tree, octets are entropy coded considering the correlation with neighboring octets [10], while isolated points are coded using the Inferred Direct Coding Mode (IDCM) [94] which infers isolated points and directly code their relative coordinates . For lossy geometry, currently, TMC13 offers two possibilities: either by just pruning the octree from the root to an arbitrary level or by combining the octree pruning with a surface approximation. A surface reconstruction approximation from the

entire octree is made using a series of triangles in a mesh-like structure, but without the connectivity information relating the triangles, this method is called *trisoup* (from triangle soup) [95], [96]. The octree is pruned at a user-defined level, then the reconstructed geometry from *trisoup* consists of refining the pruned octree with the intersections of the mesh surface and the quantization grid, thus maintaining a voxelized geometry. When *trisoup* is enabled, a mixture of octree, segment indicator, and vertex position information is sent into the geometry bitstream [10].

An attribute transfer procedure (or recoloring) to determine the attribute values for the newly reconstructed geometry is done prior to the attribute encoding. Currently in TMC13, attributes can be coded using one of the three available transforms: Region-Adaptive Hierarchical Transform (RAHT) [90], Predicting Transform [97], and Lifting Transform [98].

- **RAHT**: The RAHT is a hierarchical orthogonal subband transform [99]. It is a variation of the Haar transform, which uses adaptive weights to consider different regions with empty or occupied voxels; in order words, it takes the data's sparsity into account.

  To implement the RAHT, one must follow the backward order of the octree scan (from leaves to root), combining voxels into larger cubes until reaching the bounding cube. Occupied voxels are assigned with weight $w = 1$, and empty ones with $w = 0$. Same branch voxels have their attributes combined through a linear transformation (Eq. (6) from [90]), which roughly takes the weighted average and the weighted average difference of each voxel pair, generating one low- and one high-pass transformed coefficients, respectively. High-pass coefficients are ready to be quantized and encoded. On the other hand, the low-pass ones are promoted to the next level, and their weights are updated to the sum of the generating voxels weights. Further recombination with other low-pass coefficients is performed hierarchically and recursively. When occupied voxels are paired with empty ones, their attributes are directly promoted to the next level, but their weights do not change. Thus, densely populated regions get more importance than sparser ones in the transform. After reaching the root, the low-pass and all high-pass coefficients are quantized, and entropy coded [100]. Figure 2.21 illustrates the RAHT implementation for a $2 \times 2 \times 2$ block. Notice that differently from the

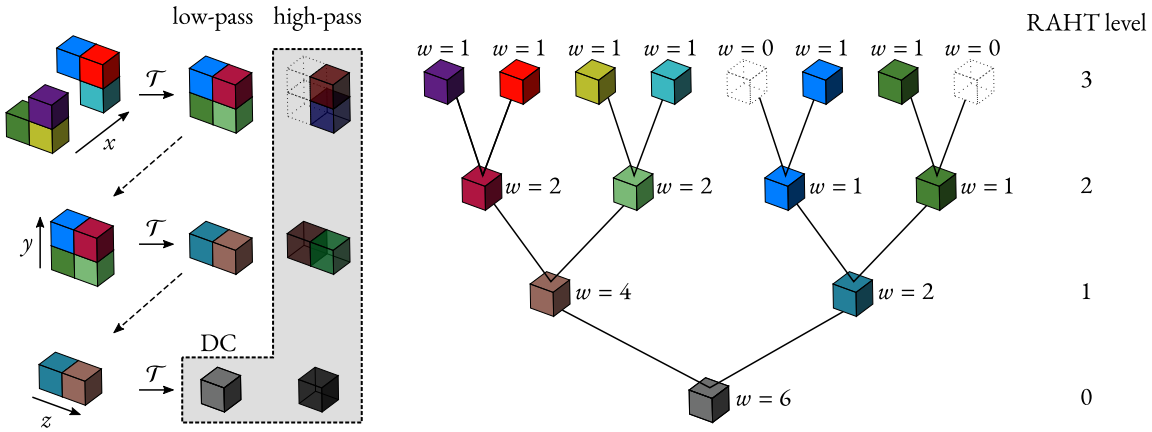


Figure 2.21: Illustration of RAHT in a $2 \times 2 \times 2$ block [10].

octree, at each level, the RAHT is performed in only one dimension; thus, one level of the octree corresponds to three levels of the RAHT.

TMC13 allows for a transform domain prediction of RAHT [101], [102]. In this approach, the RAHT octree traversal order is reversed (starting from the root and going to the leaves). Then, a step of attribute inter-depth upsampling is introduced to obtain a local prediction. This prediction is applied to high-pass coefficients so that only low-pass and prediction residual coefficients need to be encoded. The prediction of attributes is made by a weighted average of neighboring nodes. This prediction formulation of RAHT reported gains up around to 30% over RAHT without prediction [10].

- **Predicting & Lifting Transforms**: The Predicting and the Lifting Transforms follow a lifting scheme for subband decomposition [103]. In fact, the Lifting Transform is built on top of the Predicting Transform, such that both transforms are commonly referred to as *predlift*. The difference is that when using only the Predicting Transform, there is no update operator, which is commonly used for reducing the aliasing effects of the prediction step. The *predlift* forward and inverse scheme is depicted in Figure 2.22.

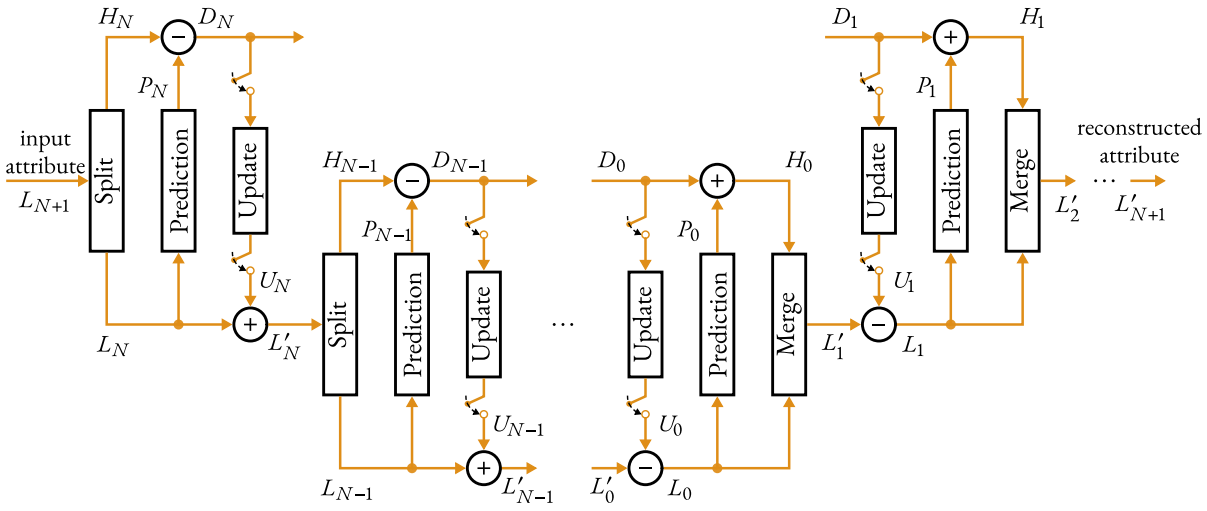


Figure 2.22: Forward and inverse *predlift* scheme [98].

The *predlift* transform relies on a Level of Details (LoD) representation, that distributes the input points into sets of refinements (R), which are created according to a set of L_1 distances specified by the user [10], [88], such that $\text{LoD}_\ell = \bigcup_{i=0}^{\ell} R_i$, as depicted in Figure 2.23

Let L_ℓ be the set of attributes associated with LoD_ℓ, and H_ℓ the set of attributes associated with R_ℓ. Thus, by the definition of LoD, $L_\ell = \bigcup_{i=0}^{\ell} H_i$. The split operator takes L_ℓ as an input and generates a low-resolution output $L_{\ell-1}$, and a high-resolution output $H_{\ell-1}$. The merge operator does the inverse, taking L_ℓ and H_ℓ as inputs, and returning $L_{\ell+1}$. The prediction operator predicts the high-resolution attributes of the points in R_ℓ, by using interpolation based on the inverse-distance weighted average of the k-nearest neighbors of R_ℓ in $\text{LoD}_{\ell-1}$ [98]. Finally, when the Lifting Transform is used, an update operator and adaptive quantization are introduced. Each point is associated with an influence weight based on its

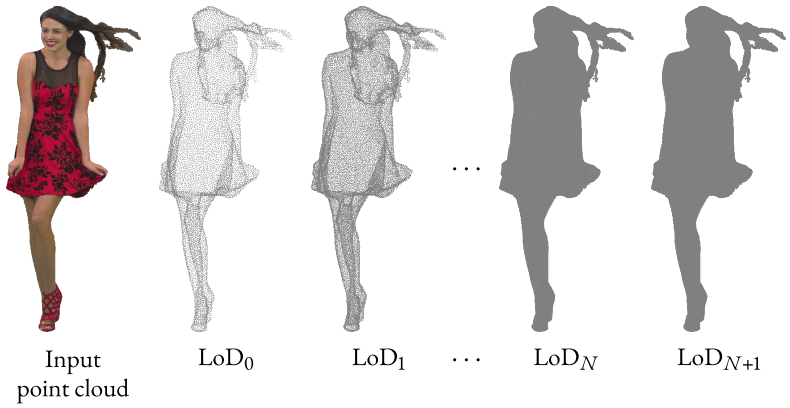


Figure 2.23: Evolution of the LoDs [98].

impact in the encoding process (points in lower LoDs have higher weights). The value of the low-resolution attributes is then updated using the prediction residuals, the distance between the predicted points and their neighbors, and their corresponding weights. Adaptive quantization is achieved by multiplying transformed coefficients by the square root of their respective weights.

2.5.2 Video-based Dynamic Mesh Compression (V-DMC)

Within MPEG, mesh compression efforts initially focused on *animated* meshes, in which vertex positions change over time, while the mesh topology, connectivity, and attributes remain constant. Solutions like Frame-based Animated Mesh Compression (FAMC) [104], [105] and tools within the Animation Framework eXtension (AFX) [106] were developed to support such cases. However, with the growing popularity of real-world captures, such as image-based reconstruction or 3D scanning, where a new mesh can be generated independently for each frame [107], [108], there emerged a need for compressing more flexible dynamic meshes, i.e., time-varying meshes (TVMs). Unlike animated meshes, TVMs allow varying topology, connectivity, and attribute sets across frames. Frames may contain different numbers of vertices, lack explicit correspondences over time, and may not even be *homeomorphic* (i.e., they do not share the same topological structure). While this flexibility makes TVMs better suited for real-world capture pipelines, it also introduces significantly greater challenges for compression,particularly in exploiting temporal coherence [29].

The compression of dynamic TVMs has been studied in the literature for over two decades [109]–[114]. Earlier methods unfolded meshes or projected them onto 2D image domains to leverage traditional video coding techniques and reduce dimensionality. To address the limitations of existing standards and better support TVM compression, MPEG 3DGH issued a CfP in 2021 [78]. Proposals were submitted by Apple [80], [115], InterDigital [116], [117], Nokia [118], [119], Tencent [120], [121], and Sony [122], [123]. Among them, Apple's solution was selected as the basis for the emerging Video-based Dynamic Mesh Compression (V-DMC) standard.

2.5.2.1 Coding Principles

In the V-DMC, the encoded data is divided into four primary components: base mesh, displacements, attributes, and atlas, as illustrated in Figure 2.24. The standard supports three coding modes: C0 for lossless compression, C1 for all-intra (AI) coding, and C2 for random access (RA) coding. The main difference between the lossy modes, C1 and C2, lies in the pre-processing and base mesh encoding stages, where mode-specific modules are activated by the encoder.

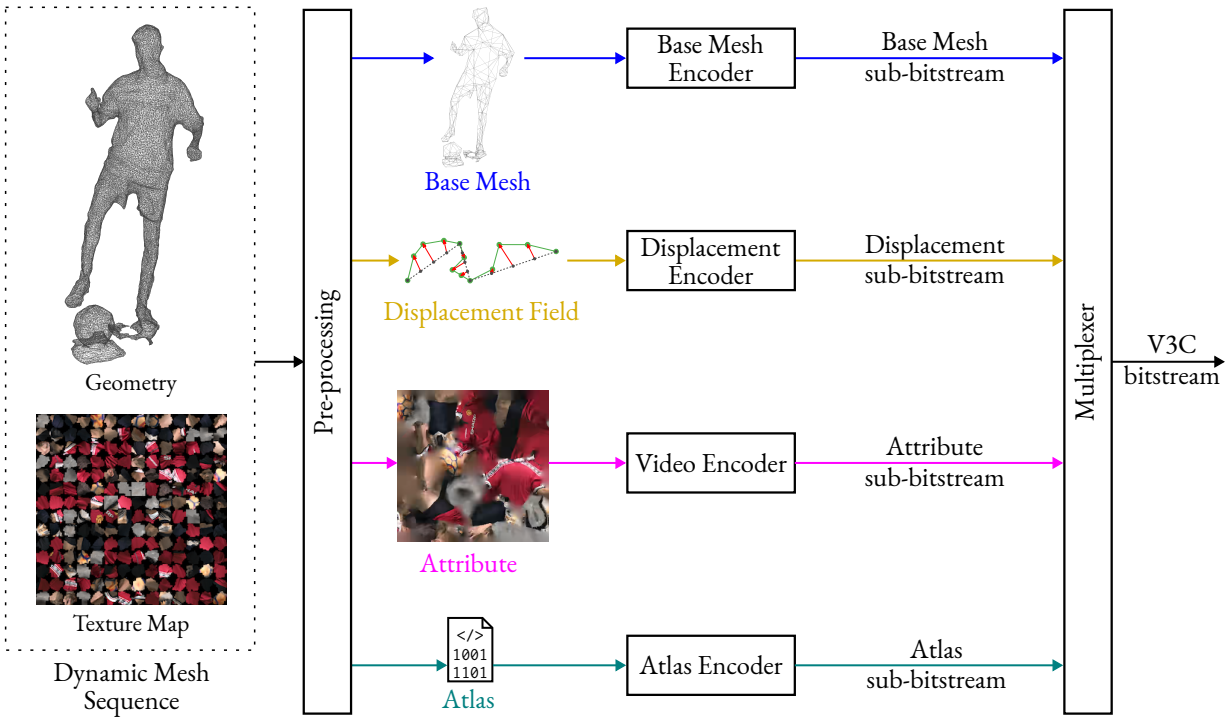


Figure 2.24: V-DMC high-level framework [11].

In the case of the lossy modes, the original mesh is approximated using a sparser base mesh, combined with a subdivision scheme [124] and a set of displacements that capture the differences in vertex positions between the original and the subdivided mesh. This representation significantly reduces the bitrate required for connectivity, since only the base mesh connectivity needs to be encoded. Furthermore, the displacements are generally small, as the subdivision process produces a mesh that closely approximates the original geometry [125], [126].

The base mesh can then be encoded using an off-the-shelf static mesh coder, in V-DMC TM this is done using MPEG Edgebreaker (MEB) [127], which is based on the original Edgebreaker [128] algorithm. Displacements undergo a wavelet transform [129], followed by quantization and then are packed into video-compatible frames, which are compressed using standard video codecs, such as HEVC. Alternatively, the quantized wavelet coefficients can be entropy coded using arithmetic coding [130].

Texture images are re-parametrized to fit the subdivided mesh, then the maps are reconstructed with temporal coherence in mind, enabling further compression using video codecs (e.g., HEVC) The atlas component contains instructions that guide the V3C bitstream [77] decoding/rendering

process, specifically detailing how the inverse reconstruction should be performed. These instructions are entropy coded using arithmetic coding.

In the lossless mode, the original mesh is encoded directly and without loss using the MEB encoder. The only pre-processing step applied is the merging of duplicate vertices [131]. In this mode, there is no displacement bitstream, and texture maps are also encoded using lossless configuration of HEVC.

2.5.2.2 Pre-processing

As shown in Figure 2.24, the pre-processing stage in V-DMC is really important, as it dictates the quality of the base mesh, the creation of texture coordinates, and the distribution of displacements, all of which define what actually gets encoded, thus influencing the final quality of the reconstructed mesh and the compression efficiency. The pre-processing pipeline includes three sub-blocks: (1) Decimation and Mapping, (2) Texture Parameterization, and (3) Subdivision Surface Fitting, as shown in Figure 2.25. The goal of this stage is to generate a base mesh that, after the subdivision, approximates the original mesh as closely as possible.

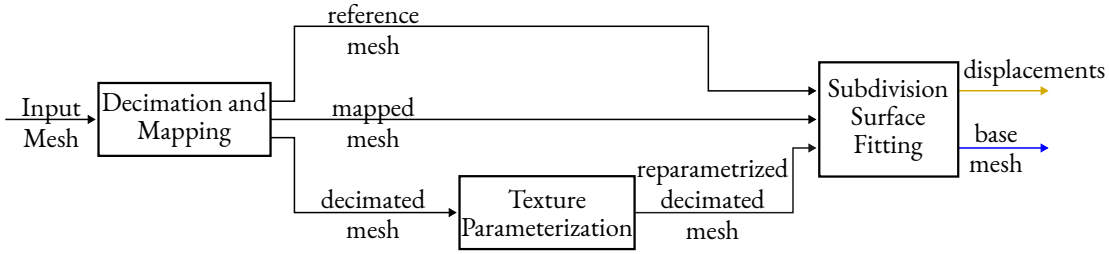


Figure 2.25: V-DMC pre-processing stage.

In the first sub-block, three versions are derived from the original mesh: a *reference mesh*, obtained by merging duplicate vertices in the input; a *decimated mesh*, which is a simplified version of the reference mesh; and a *mapped mesh*, where the reference mesh is deformed to match the geometry of the decimated version. Decimation is done using a quadric error metric surface simplification technique, which employs an edge-collapsing strategy to reduce the number of triangles while preserving the overall shape of the mesh [132].

The decimated mesh is then reparameterized using the UVAtlas tool [133], producing a compact set of texture patches that can reduce discontinuities and improve RD performance. The current TM also includes an alternative parameterization method, orthoAtlas [134], which uses an orthogonal projection strategy similar to V-PCC [10]. An example of the texture parameterization is shown in Figure 2.26. Although this step does not change the number of triangles, it may increase the vertex count, since reparameterization can introduce additional vertices to better align with texture patch boundaries.

In the subdivision surface fitting module, the reparameterized decimated mesh is first subdivided. Each vertex of the subdivided mesh then searches for its nearest point on the mapped mesh, which is used as an intermediate reference since its geometry closely matches that of the subdivided

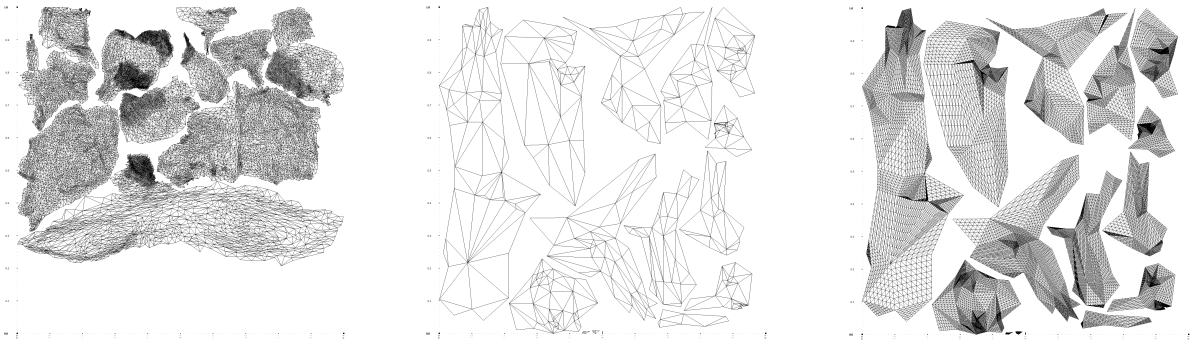


Figure 2.26: Texture parameterization. It is possible to see the reduction in the number of patches from the the original map (a) to the reparameterized map for the decimated mesh in (b). In (c), the reparameterized map for the subdivided mesh, notice that the patches remain the same, but the number of triangles has increased.

mesh, improving the accuracy of the nearest-neighbor search. Once the nearest points are located on the mapped mesh, their corresponding vertices in the reference mesh (which shares the same vertex set as the mapped mesh) are used to compute the displacement field as the differences between the 3D positions of the subdivided mesh vertices and their reference mesh counterparts. Finally, the base mesh is obtained by decimating the subdivided mesh after a feedback loop that iteratively adjusts vertex positions to minimize the distance between the subdivided mesh and its displaced version [80], [135]. The overall process is illustrated in Figure 2.27 and in Figure 2.26.

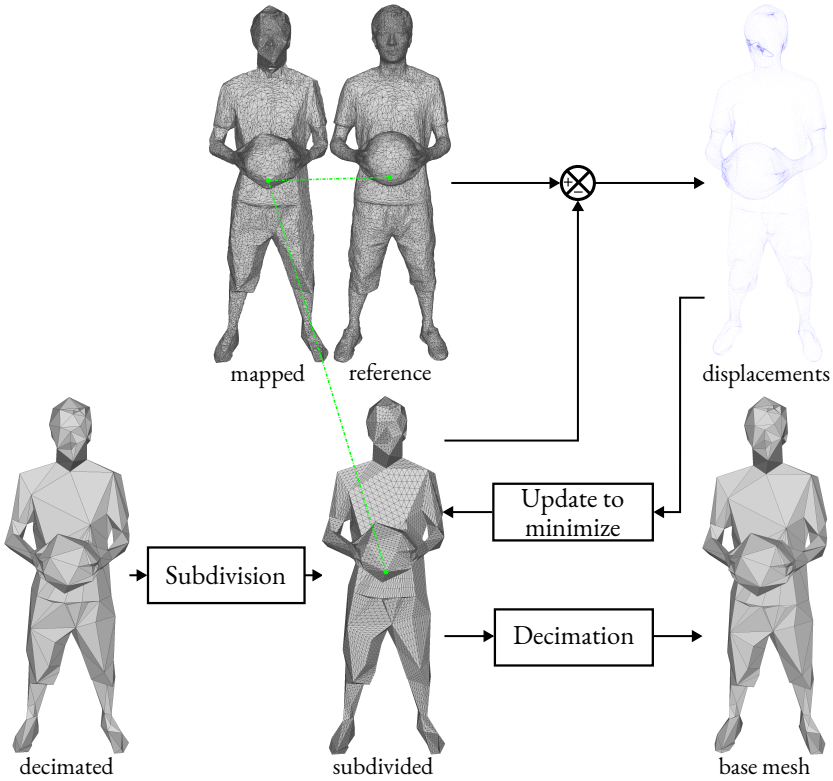


Figure 2.27: Illustration of the subdivision surface fitting.

In V-DMC, any subdivision scheme may be employed, provided it can be exactly replicated at

the decoder. The simplest option is midpoint subdivision, in which each edge is split into two equal segments by inserting a vertex at its midpoint. With each iteration, every triangle is divided into four smaller sub-triangles, as illustrated in Figure 2.27 and in Figure 2.26, for geometry and texture coordinates, respectively. This iterative refinement produces a hierarchy of meshes at different LoDs, where LoD_0 corresponds to the coarsest base mesh and higher LoDs progressively add geometric detail. Ongoing work explores more advanced schemes, such as Loop subdivision [136], as well as combining different subdivision strategies at each iteration.

When operating in C2 mode (inter-frame coding), a temporally consistent re-meshing can be achieved during pre-processing by reusing the same base mesh across consecutive frames. This allows the current mesh to be subdivided using the same subdivision structure as the previous frame, thus enabling the use of a single base mesh for multiple frames while transmitting the corresponding motion vectors. Note that such time-consistent re-meshing is not always feasible [80].

2.5.2.3 Codec Architecture

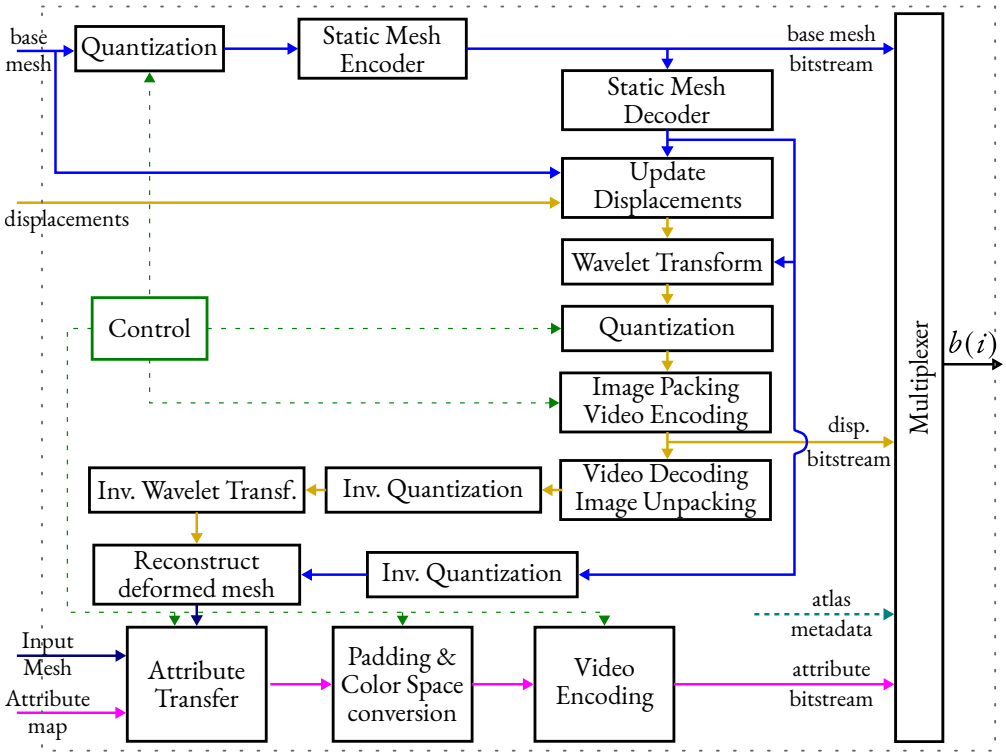


Figure 2.28: V-DMC encoder.

After the pre-processing stage, the encoding process begins, as illustrated in Figure 2.28. The base mesh is first quantized, then encoded using the MEB encoder, producing the base mesh sub-bitstream. Not shown in Figure 2.28, but in C2 mode, when the base mesh is reused for multiple frames, motion field coding is performed instead of the base mesh encoding [80]. Since this encoding can slightly change the base mesh geometry, the base mesh is then decoded and the displacements are updated accordingly. In C2, the motion field is also decoded and applied to the base

mesh, for displacement update.

The displacements are then transformed using a linear wavelet transform [137], [138], which predicts from coarser levels and encodes only the residual differences. The resulting coefficients are quantized, and packed into video-compatible frames, as depicted in Figure 2.29. Optionally, the quantized coefficients can be entropy coded using arithmetic coding [130] (not in the diagram).

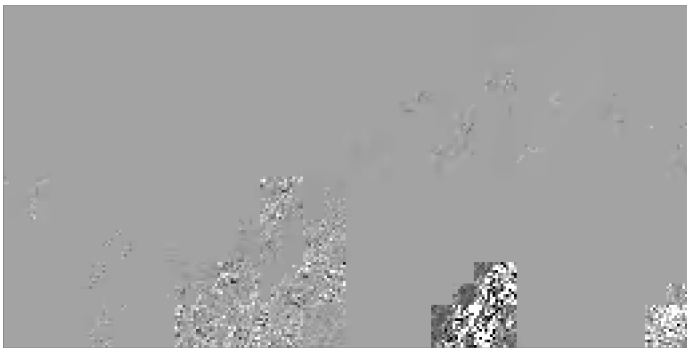


Figure 2.29: Packed wavelet coefficients of the displacements into a video frame. Gray regions represent zero-valued coefficients, while black and white represent positive and negative coefficients. The coefficients from each LoD are arranged in a block-aligned packing scheme, with ascending LoDs placed from left to right and bottom to top. Gray padding blocks are inserted both to maintain alignment with block boundaries and to fill the frame to its full size.

For the attribute coding, first, the original texture is transferred to the new parameterization from the pre-processing stage, which was updated to match the reconstructed subdivided mesh (deformed mesh). For each pixel in the output map, the method checks whether its texture coordinates fall within a valid triangle in the reconstructed mesh. If so, it finds the corresponding point on the original mesh via barycentric interpolation and nearest-neighbor search in 3D, then samples the original texture map to obtain the attribute value. If not, the pixel is marked as empty. Empty pixels are subsequently filled using a combination of the pull-push [139] algorithm and sparse linear padding [140]. The process is illustrated in Figure 2.30. After transfer and padding, the images are converted to YCbCr [44] color space and compressed using a standard video codec.

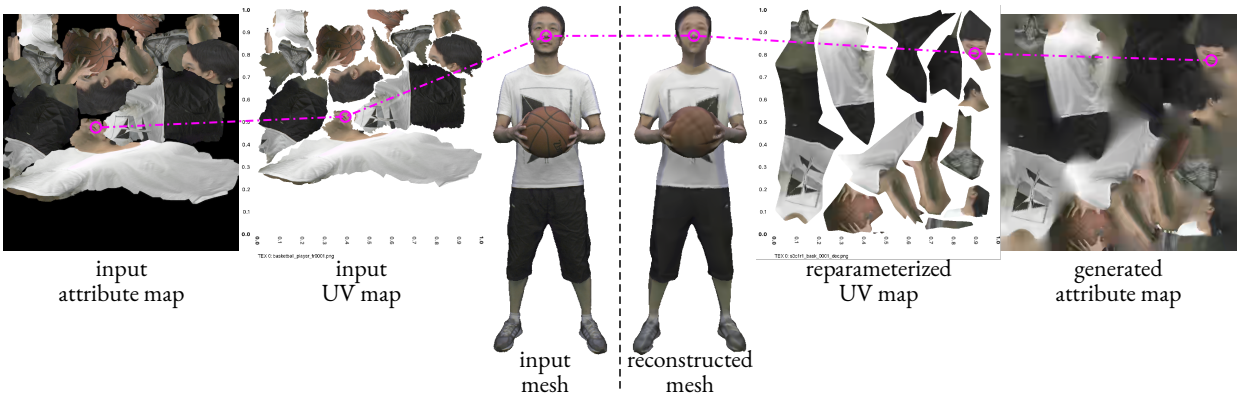


Figure 2.30: Attribute transfer and map generation [80]. Illustration of how a pixel in the image domain is generated from the UV domain, going through 3D nearest-neighbor search and barycentric interpolation back to the UV domain to find the corresponding point in the original attribute image. The empty pixels are filled using padding techniques.

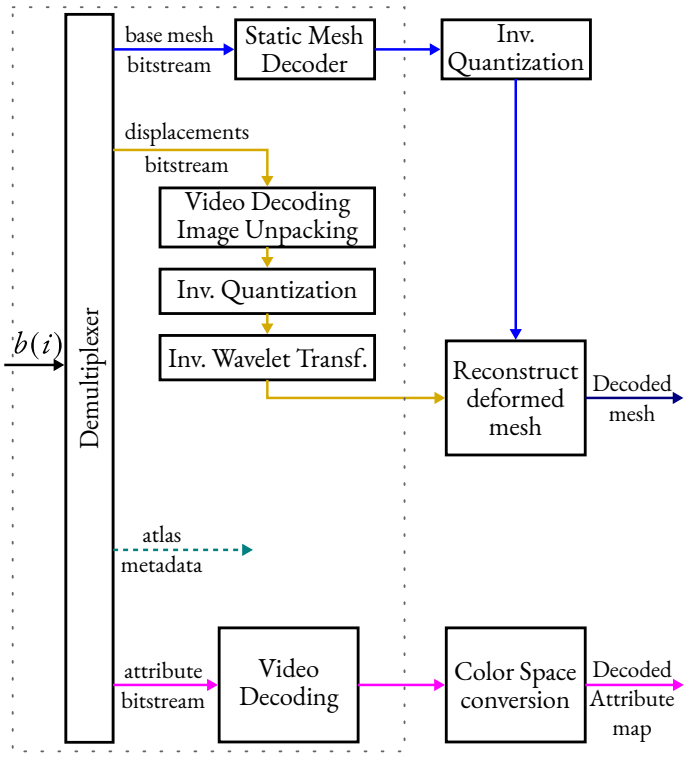


Figure 2.31: V-DMC decoder.

The decoding process follows the reverse order of the encoding steps, as illustrated in Figure 2.31. First, the base mesh is reconstructed using the MEB decoder, then inverse-quantized and subdivided. At the same time, the wavelet coefficients are unpacked from the video frames, inverse-quantized, and transformed back into displacements through the inverse wavelet transform. These displacements are applied to the subdivided mesh to recover the original geometry. Finally, the texture maps are decoded from the video frames and converted back to the RGB color space.

2.6 SCALABLE CODING

Some aspects of scalable coding are not always clearly defined, particularly when the concept is applied across different media types such as images, video, and 3D content. This section provides a concise overview of scalable coding, highlighting the principles most relevant to this work.

Scalable coding is a paradigm in which a single compressed bitstream is structured so that subsets of the stream can be decoded into valid reconstructions of the source at reduced quality, resolution, or fidelity, while full decoding yields the highest quality version. This avoids the need to encode and transmit multiple independent streams for different conditions, thereby reducing storage and bandwidth requirements. Its usefulness lies in the ability to adapt seamlessly to heterogeneous receivers operating under diverse network, hardware, or display constraints: from mobile devices on low-bandwidth connections to high-performance systems with large displays. In such scenarios, each client can consume only the portion of the bitstream suited to its own capabilities,

while all share a common source encoding. The principle is exemplified in Figure 2.32, where a single bitstream can be decoded at some resolution appropriate to the spatial requirements of the rendered object.

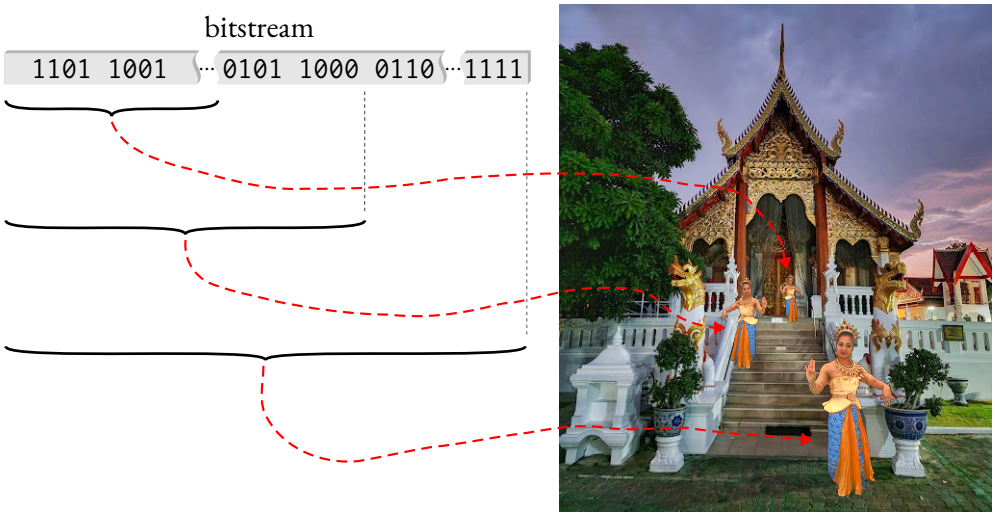


Figure 2.32: Scalable coding principle. Subsequent portions of the same bitstream are used to add details and resolution to the reconstructed 3D model. The multiple instances are shown only to illustrate how varying the decoded bitstream length affects reconstruction quality; in practice, a single object would be decoded for a defined position [141].

Although not strictly required in a coding system, scalability is often a highly desirable property as it provides flexibility, extends the applicability of a codec, and enhances efficiency in practical deployment scenarios. The main challenge of scalability is that it often comes at the cost of coding efficiency and/or increased complexity, which must be balanced against the flexibility and adaptability that this property offers.

There exist different types of scalability, regarding which aspect of the media is enhanced as more bits are decoded, such as:

- Temporal scalability: allows for varying frame rates;
- Spatial scalability: enables different resolutions;
- Quality scalability (also known as fidelity or SNR scalability): supports different quality levels at the same resolution;
- Region-of-interest (ROI) scalability: focuses on specific areas within the content;
- Object-based scalability: deals with individual objects or layers.

The different types of scalability can also be combined, enabling a single scalable bitstream to provide many representations with various spatial and temporal resolutions, quality levels, and bitrates.

Different mechanisms can be employed to achieve scalability, depending on the type of media and the desired trade-offs. In *layered coding*, the bitstream is explicitly structured into a base layer and one or more enhancement layers. Each enhancement layer refines a particular aspect of the

content (e.g., temporal resolution, spatial resolution, or quality) by reusing information from the base layer. This approach provides coarser adaptation steps but offers strong interoperability, since the base layer can often be decoded independently by legacy decoders. Layered coding has been widely adopted in video coding standards such as AVC/SVC [142], SHEVC [143], and VVC [144].

In contrast, *progressive coding* organizes the bitstream as a single refinement stream, so that decoding more bits directly improves the fidelity of the reconstruction. This allows for very fine-grained adaptation to the available bitrate, since the stream can be truncated at arbitrary points while still yielding the best possible reconstruction for the number of decoded bits. However, this flexibility comes at the cost of requiring specific decoder support, as there is no explicit layer separation. Progressive coding is commonly used in still image compression, especially with transform-based subband methods such as the Embedded Zerotree Wavelet (EZW) [145], Set Partitioning in Hierarchical Trees (SPIHT) [146], and JPEG2000 [147], with its embedded block coding with optimized truncation (EBCOT) [148] coding engine. A special case of progressive coding is *embedded coding*, where every prefix of the bitstream forms a valid code that produces a complete reconstruction at the corresponding quality level. This property allows fine-grained adaptation of the rate-distortion trade-off, since the stream can be truncated at any point while still providing the best possible reconstruction for the available bitrate.

In point clouds, the octree structure used to code geometry (see Section 2.5.1) is inherently scalable because of how the subdivision of the 3D space is done. Scalability in attributes is not as direct because the geometry is needed for attribute coding/decoding [141], [149], [150]. In meshes, progressive coding is achieved by exploiting different spatial resolutions at different LoDs, where each decoded LoD provides more details to refine the reconstruction (Figure 2.33).

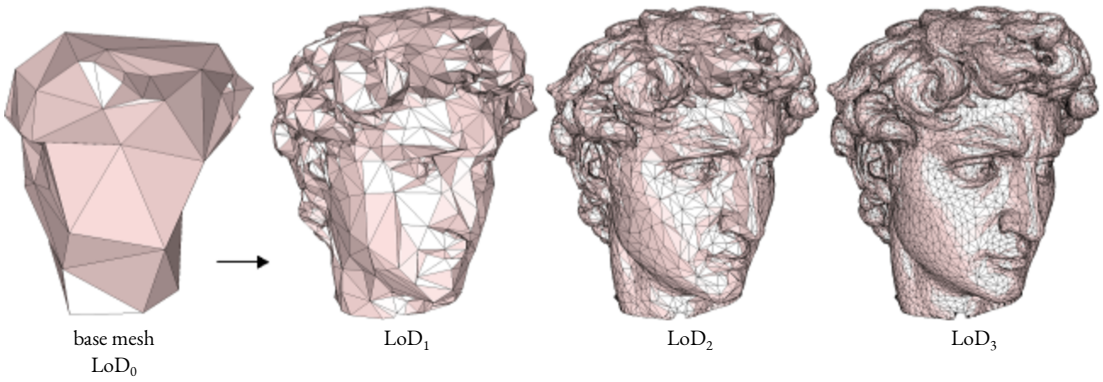


Figure 2.33: Progressive decoding of mesh geometry using LoDs [30].

An important aspect of the work developed in Chapter 4 is related to principles used in the EZW algorithm for images. Next, a brief summary of set partitioning and zerotree coding is presented.

2.6.1 Set Partition Coding and the Embedded Zerotree Wavelet (EZW)

The central idea behind set partition coding methods is to exploit the fact that groups of samples can often be represented with fewer bits than if each sample were coded individually. In raw representation, each sample must be stored with enough bits to cover the full dynamic range (e.g., 8 bits per pixel for an image with values in $[0, 255]$). However, if a group of samples is known to have values that never exceed a much smaller threshold (e.g., 15), then only $\log_2 16 = 4$ bits per sample are needed. Thus, compression is achieved whenever the bit savings from reduced representation exceed the overhead required to signal the group's location and threshold.

This way of coding is fundamentally lossless, but unlike other general-purpose entropy coders such as Huffman [151] or arithmetic coding [152], which can be used with any type of data, the efficiency of set partition coding depends on the presence of large groups of samples with small maximum values. Natural images rarely have this property, but it becomes common after applying transforms such as the discrete cosine transform (DCT) or the discrete wavelet transform (DWT), where most coefficients are small in magnitude and only a few carry significant energy. Set partition coding methods are, thus, typically applied to transformed coefficients, where they can adaptively organize data into groups of different amplitude ranges [153].

One way of partitioning the data is using hierarchical trees linking the coefficients. The DWT is especially suitable for this tree-structure, as it naturally organizes coefficients into subbands with distinct spatial frequency ranges. These so called spatial orientation trees (SOT) branch successively to higher frequency subbands at the same spatial orientation. An example of a single SOT in the wavelet transform domain is shown in Figure 2.34(a). Note that the roots are the coefficients in the lowest frequency subband and branch by three to a coefficient in the same relative position in each of the other three lowest level subbands. All subsequent branching of each node is to four adjacent coefficients in the next higher level subband in the same relative position. The SOTs are non-overlapping and taken altogether include all the transform coefficients. Therefore, the set of all SOTs partitions the wavelet transform.

The EZW [145] is an embedded SOT coding method, which is not strictly a set partition coder, as it locates only significant coefficients and trees, all of whose coefficients, including the root are insignificant. When the thresholds are integer powers of 2, insignificant trees mean that all the bits in the corresponding bit-plane located at the coordinates of the nodes are 0's. Hence their term *zerotree*. SPIHT [146] builds upon the EZW by locating the zerotrees in a broader sense, also considering the spatial redundancy in the transformed domain, allowing for more efficient coding of the significant coefficients.

The EZW algorithm exploits statistical redundancy in the hierarchical structure of the SOTs. Given a significance threshold T_n, each coefficient and its descendants are tested for significance. Subtrees in which all coefficients (including the root) are insignificant are identified as *zerotrees*, and their root coefficients are encoded using the *zerotree root* (ZTR) symbol. When the root itself is insignificant but at least one descendant is significant, the root is coded as an isolated zero (IZ). If the root is significant, it is marked as positive (POS) or negative (NEG), and its descendants are

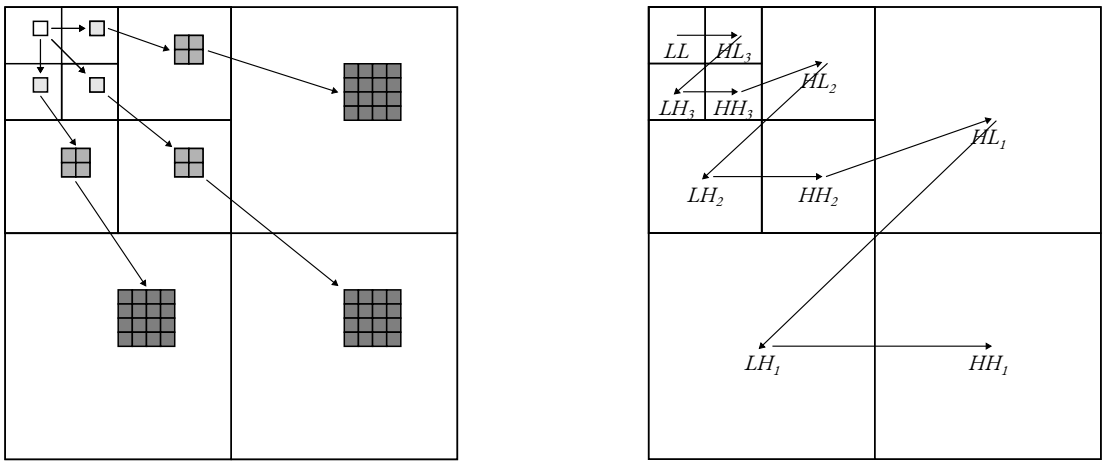


Figure 2.34: Example of a 3-level wavelet decomposition scheme. (a) A spatial orientation tree of a discrete wavelet transform. (b) Scanning order of subbands in the EZW algorithm. The coefficients are scanned in a zig-zag order, starting from the lowest frequency subband (LL) and going to the highest frequency subbands. Subbands are named for horizontal and vertical low-(L) or high-pass band (H), e.g., LH_1 is horizontal low-pass band and vertical high-pass band at first recursion level.

not characterized at this threshold.

When a coefficient has no descendants (i.e., it belongs to a level-1 subband or is a leaf node), an insignificant value is encoded with a dedicated zero symbol (Z). Since both encoder and decoder know when the leaf node set is reached, they can switch to a reduced symbol alphabet. In this case, the alphabet becomes {Z, POS, NEG}, mapped to the raw bit codes `0`, `11`, and `10`, respectively, thereby saving one bit whenever an insignificant leaf node is encountered.

Formally, let a tree (or subtree) rooted at coefficient $w_{(i,j)}$ with descendant set $\mathcal{D}_{(i,j)}$ be

$$\mathcal{T}_{(i,j)} = \{w_{(i,j)}\} \cup \mathcal{D}_{(i,j)}. \tag{2.37}$$

A subtree is considered significant if at least one of its coefficient exceeds the threshold:

$$S_n(\mathcal{T}_{(i,j)}) = \begin{cases} 1, & \text{if } \max\{|w_{(k,l)}| : w_{(k,l)} \in \mathcal{T}_{(i,j)}\} \geq T_n \\ 0, & \text{otherwise.} \end{cases} \tag{2.38}$$

Depending on the significance of the root and its subtree, one of four symbols is output:

$S_n(\mathcal{T}_{(i,j)})$	$S_n(w_{(i,j)})$	$w_{(i,j)} > 0$	output symbol
0	0	–	ZTR,
1	0	–	IZ,
1	1	yes	POS,
1	1	no	NEG.

(2.39)

For leaf-nodes, i.e., $\mathcal{D}_{(i,j)} = \emptyset$, then, $\mathcal{T}_{(i,j)} = \{w_{(i,j)}\}$, the alphabet is reduced to

$S_n(w_{(i,j)})$	$w_{(i,j)} > 0$	output symbol
0	–	Z,
1	yes	POS,
1	no	NEG.

(2.40)

The algorithm maintains two ordered control lists: the dominant list (DL), and the subordinate list (SL). The DL holds coordinates of subtree roots to be tested at the current threshold, while the SL stores coordinates of coefficients already found significant, to be refined later. Coefficients are visited in a predefined scan order (see Figure 2.34(b)), typically zigzagging across subbands to exploit spatial-frequency correlations. After each iteration, the threshold is halved ($T_{n+1} = T_n/2$), and the process repeats until either all bit-planes are exhausted or the bit budget is met. The pseudocode in Algorithm 1 describes the overall coding process.

In this way, EZW produces an embedded bitstream, where decoding can be truncated at any point to obtain a valid reconstruction at reduced fidelity, making it highly efficient for progressive transmission and scalable coding. More details on EZW, SPIHT and set partition coding can be found in [153].

Algorithm 1: Embedded Zerotree Wavelet (EZW)

$T_0 = \lfloor \log_2 \{ \max |w_{(i,j)}| \} \rfloor$
$DL = [w_{(i,j)}$ **for** (i, j) **in** lowest-frequency subband$]$ `// LL from Fig. 2.34(b)`
$SL = []$
$n = 0$
$R_b =$ `bit_budget`
`bitstream = ''`
`/* Scanning order follows the zig-zag of Fig. 2.34(b).`
`Coordinates within a subband are visited in raster scan order from top to bottom. */`
while $(len($`bitstream`$) < R_b)$ **and** $(T_n > 0)$ **do**
    Dominant pass (threshold T_n**):**
    foreach $w_{(i,j)}$ **in** DL **do**
        if $S_n(\mathcal{T}_{(i,j)}) = 1$ **then** `// IZ, POS or NEG`
            Append offspring (if existent) to DL
            if $S_n(w_{(i,j)}) = 1$ **then** `// POS or NEG`
                Append $w_{(i,j)}$ to SL
                if $w_{(i,j)} > 0$ **then** `// POS`
                    Append '`11`' to `bitstream`
                else `// NEG`
                    Append '`10`' to `bitstream`
            else `// IZ`
                Append '`01`' to `bitstream`
        else `// ZTR or Z`
            if $w_{(i,j)}$ **is not** leaf **then**
                Append '`00`' to `bitstream`
            else
                Append '`0`' to `bitstream`
    Subordinate pass (refinement at T_n**):**
    foreach $w_{(i,j)}$ **in** SL, except those included in the last Dominant pass **do**
        Append the k-th most significant bit of $|w_{(i,j)}|$ to `bitstream`
    Threshold update:
    $T_{n+1} \leftarrow T_n/2$
    $n \leftarrow n+1$

3 FRACTIONAL SUPER-RESOLUTION OF VOXELIZED POINT CLOUDS AND INTERPOLATIVE COMPRESSION

As discussed in Section 2.5, G-PCC operates on *voxelized* point cloud representations, imposing a structured grid-based format to enable efficient compression. In practice, this representation often leads to *downsampled* or low-resolution (LR) versions of the original point clouds, as geometric detail is reduced to meet bitrate constraints. Such degradation can significantly affect applications that depend on high-fidelity 3D reconstructions. To mitigate this problem, *super-resolution* (SR) methods aim to reconstruct high-resolution (HR) point clouds from their compressed LR counterparts, restoring fine geometric structures and improving both visual fidelity and structural accuracy.

In this chapter, we present a super-resolution method for voxelized point clouds that takes a grid-downsampled LR point cloud as input and leverages the inherent grid constraints, density variations, and self-similarities to generate a grid-super-resolved version of the input. This method was initially developed in [12], building upon the work of Garcia *et al.* [154].

Grid resampling (see Section 3.2) for voxelized point clouds was an underexplored topic in the point cloud super-resolution literature at the time this method was developed. Moreover, as will be presented later in this chapter, the proposed method was designed with the octree representation in mind,widely used in PCC, making it naturally suited for interpolative compression. This methodological approach was developed within the evolving landscape of point cloud compression and remains relevant today.

3.1 RELATED WORK

Numerous approaches have been proposed for point cloud SR, but direct comparisons are often challenging due to variations in the number of inputs, differences in LR versions, or the use of extra information.

Optimization-based Methods

In optimization-based point cloud SR, a cost function is defined and new points are added to minimize this function. Alexa *et al.* [155] constructed a Voronoi diagram [156] on the 3D surface and insert points at the vertices to minimize a moving least squares (MLS) cost function. Subsequent works employing the MLS cost function [157], [158] generally tend to over-smooth the geometry. To address this, Huang *et al.* [159] introduced an edge-aware solution that relies on accurate normal estimation and careful parameter tuning to mitigate over-smoothing.

Similarly, Hamdi-Cherif *et al.* [160] exploited the similarity of local descriptors for point cloud SR, though their method required multiple scans, explicit normal calculations, and assumed surface smoothness. Dinesh *et al.* [161], [162] employed Delaunay triangulation [163] on the LR point cloud and optimized an L_1-norm graph-total-variation (GTV) cost function for neighborhood surface normals, promoting piecewise smoothness in reconstructed 2D surfaces under the constraint that LR coordinates remain unchanged. It is worth noting that these optimization-based methods were primarily developed for static, non-voxelized point clouds and are typically categorized as *set upsamplings*. Although Dinesh *et al.* [162] evaluated their method on a voxelized point cloud, it remains unclear whether the SR output maintained voxelization.

Deep Learning-based Methods

Deep learning methods have also been explored for point cloud SR, starting with PU-NET by Yu *et al.* [164], which learns multi-scale features by downsampling the input and reconstructing a denser point set using multi-branch multilayer perceptrons (MLPs). Later, Yu *et al.* proposed EC-NET [165], an edge-aware network designed for point consolidation, though its training process required costly edge annotations. Other works, such as 3PU by Wang *et al.* [166], introduced a progressive upsampling strategy to reduce noise and preserve fine geometric details, albeit with high computational costs and a substantial training data requirement.

Further advancements include PU-GAN [167], which applied adversarial learning to enhance uniformity in SR results, with performance improvements primarily driven by the discriminator network. Graph convolutional networks (GCNs) were also investigated for point cloud SR by Wu *et al.* [168] and Qian *et al.* [169]. PUGeo-Net [170] proposed an SR method that leverages the first and second fundamental forms to model local geometry, although it required normal vectors for inference.

A key limitation of deep learning-based approaches is their reliance on fixed upsampling scales, necessitating retraining when different scale factors are needed, making them impractical for dynamic applications. Recently, Ye *et al.* [171] introduced Meta-PU, a meta-learning framework that supports arbitrary-scale upsampling without retraining by dynamically adjusting network weights for different scaling factors. However, data-driven methods are often constrained by the specific upsampling types and point cloud formats used during training, requiring adjustments when applied to new formats. Most of these methods focus on static, non-voxelized point clouds, with some extending to LiDAR data [167], [169], [170].

Voxelized- and Octree-based Methods

Previous methods did not account for voxelization, raising concerns about their applicability to standards such as G-PCC, which relies on a structured octree-based representation. Octree-based point cloud SR methods naturally accommodate such constraints by leveraging spatial correlations within the octree structure to reconstruct missing details. These approaches typically expand

pruned branches in the octree, improving precision and adding points to the grid-based representation.

Garcia *et al.* developed two SR methods for dynamic point clouds, both leveraging statistical data gathered from previous frames of the sequence: SR by example [172] and SR by neighborhood inheritance [154]. The first explores similarities between time-adjacent frames to predict voxels at higher levels of the octree. The LR frame is super-resolved using the similarities from the previous frame at full-resolution. The second creates a dictionary of child nodes based on the neighborhood configuration from previous full-resolution frames. Then, the neighborhood from each occupied voxel is used to estimate its child nodes.

Proposed Method

The approach presented in this chapter extends the neighborhood inheritance SR method [154], eliminating the dependency on multiple point clouds for dictionary creation (intra-frame SR) and improving fractional resampling capabilities, allowing for arbitrary upsampling scales rather than being constrained to powers of two.

3.2 POINT CLOUD RESAMPLING

Point cloud resampling plays a crucial role in compression, rendering, and reconstruction by adjusting the resolution and structure of the data. The choice of resampling strategy can have a significant impact on the coding efficiency, spatial correlation, and quality of the decoded point cloud.

3.2.1 Downsampling

Point cloud downsampling can be broadly categorized into *set downsampling* and *grid downsampling*, each with distinct characteristics and implications for compression and reconstruction. In set downsampling, points are removed usually on a distance-based criterion, such as Poisson disk sampling [173]. This approach prioritizes evenly spaced density reduction, this alters the frequency distribution, rather than uniformly removing high-frequency details. As a result, some fine details may remain, but at the cost of introducing holes, isolated points, and disrupting spatial correlation, ultimately reducing coding efficiency in compression.

Grid downsampling, in contrast, is performed by (re)voxelization, where points are mapped onto a structured grid and merged if they fall within the same voxel. This approach reduces resolution while increasing local density compared to the original point cloud, leading to blocky artifacts in the geometry (loss of high-frequency details). Once a point cloud is downsampled using voxelization, the remaining LR points cannot be rescaled to their original resolution without introducing

errors.

Figure 3.1 illustrates both set downsampling and grid downsampling, where each method is used to reduce the number of points to a similar level. To facilitate comparison, the grid-downsampled point cloud is expanded and visualized with larger voxel sizes. The $D1_{PSNR}$ distortion metric, detailed in Section 2.4, quantifies the geometric differences between the downsampled versions and the original point cloud.

Figure 3.1: Comparison of downsampling approaches. Original point cloud (857,966 voxels), set downsampling (64,176 voxels $D1_{PSNR}$ = 61.5dB), and grid downsampling (62,130 voxels, $D1_{PSNR}$ = 58.4dB).

Grid downsampling can be formally expressed by scaling the geometry V by a scale factor $s > 1$ and rounding the coordinates to align with an integer grid. Duplicate points are merged, and their attributes (e.g., color) are averaged. The resulting downsampled geometry V_d is given by

$$V_d = \texttt{unique}\left(\texttt{round}\left(\frac{V}{s}\right)\right), \tag{3.1}$$

where $\texttt{unique}(X)$ returns the unique vectors in the set X, and $\texttt{round}(\cdot)$ rounds the vector components to the nearest integer. The consolidation of duplicate position information is similar to the voxelization process.

The relationship between V_d and V can be viewed as a hierarchical tree, where V_d represents parent nodes and V represents child nodes. Figure 3.2 illustrates this concept in one dimension (1D). When s is an integer, each parent node has an equal number of child nodes, leading to a regular downsampling process. However, when s is not an integer, the number of child nodes per parent varies, depending the parent node's coordinates. In cases where $1 < s < 2$, some parent nodes may have only one child (uniparous), while others have two children (multiparous).

In 3D downsampling, integer values of s result in a regular hierarchical structure, where each group of $s \times s \times s$ voxels in V is reduced to a single voxel in V_d, as shown in Figure 3.3(a). In this case, each parent node has s^3 children, effectively representing a pruning operation in the octree structure. When $s = 2^n, n = 1, 2, 3, \ldots$, pruning occurs at level $d-n$. Note that, to achieve a strictly exact pruning corresponding with the octree level $d - n$, the round() function in Equation (3.1)

(a) $s = 2$ (b) $s = 1.25$

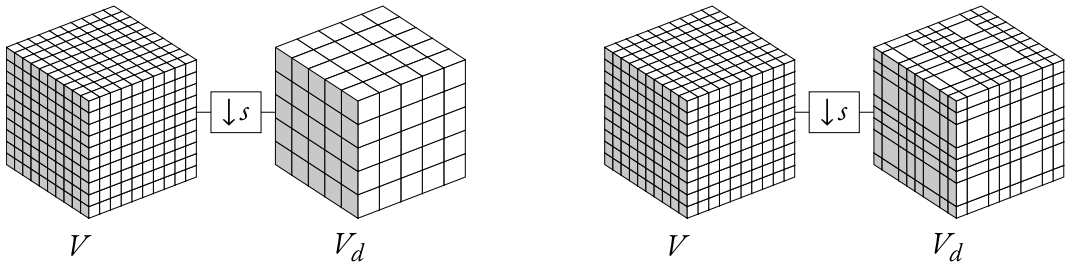
Figure 3.2: Illustration of the downsampling process in a 1D voxelized structure. Parent nodes (x_d) serve as hierarchical representatives of child nodes (x).

must be replaced by `floor()`.

However, when s is not an integer, each parent node in V_d may have a varying number of children, leading to an irregular voxel grid. As illustrated in Figure 3.3(b), if $1 < s \leq 2$, a parent node may have 1, 2, 4, or 8 children, depending on its spatial position. The number of child nodes per parent can be generalized as

$$i_{\max} = (\lceil s \rceil - 1)^u \lceil s \rceil^m, \tag{3.2}$$

where u is the number of uniparous coordinates, m is the number of multiparous coordinates, and $\lceil\ \rceil$ denotes the ceiling function. Although non-integer values of s create irregular voxel grids, patterns emerge when s is expressed as a rational fraction p/q (where $p > q$). This concept is referred to as fractional resampling, which enables flexible downsampling and upsampling beyond standard integer scale factors.



(a) $s = 2$ (b) $s = 1.25$

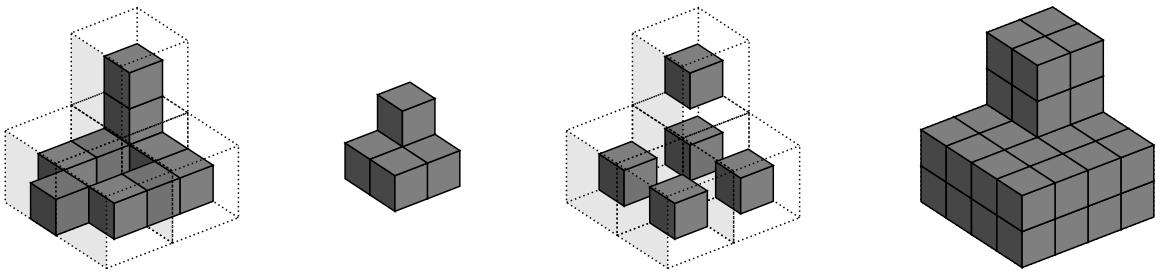
Figure 3.3: Downsampling in a voxel grid. When using a fractional s value (b), parent-child relationships vary.

3.2.2 Upsampling

We define point cloud grid upsampling as the inverse operation of the previously described grid downsampling process. The bounding volume containing the voxelized point cloud is requantized, this time to increase spatial resolution. A straightforward approach to upsampling is to simply expand the downsampled geometry by scaling it back to the original resolution

$$V_e = \mathsf{round}\,(V_d \cdot s)\,. \tag{3.3}$$

However, this method results in a sparse point cloud, as it does not recover the missing points lost during downsampling. To generate a denser and more visually consistent reconstruction, interpolation is required to populate the missing voxels.



(a) V (b) V_d (c) $V_e = s \cdot V_d$ (d) V_u

Figure 3.4: Illustration of the NNI upsampling for $s = 2$: (a) child nodes (V) surrounded by its parent nodes; (b) parent nodes (V_d); (c) expanded geometry; (d) NNI upsampling.

➛ NEAREST-NEIGHBOR INTERPOLATION

The baseline technique for geometry upsampling is the nearest-neighbor interpolation (NNI), which sets all children from the parent nodes in V_d as occupied. The texture upsampling for the NNI usually follows the same idea used for the geometry: the colors from parent nodes are just replicated to their corresponding children.

Figure 3.4 illustrates the upsampling process:

1. A point cloud V is downsampled, producing a low-resolution representation V_d.
2. The downsampled points are expanded to a higher resolution, resulting in V_e, a sparse version of the original point cloud.
3. The missing points are interpolated using NNI, yielding the final upsampled version V_u.

The NNI upsampling process ensures that all child nodes $\{\mathbf{v}_u(i)\}$ from a parent voxel $\mathbf{v}_d(k)$ satisfy

$$\mathbf{v}_d(k) = \texttt{round}(\mathbf{v}_u(i)/s), \tag{3.4}$$

for every $i = 1, 2, \ldots, i_{\max}$. Conversely, the upsampled voxels are computed as

$$\mathbf{v}_u(i) = \texttt{round}(s \cdot \mathbf{v}_d(k)) + \epsilon(i), \tag{3.5}$$

where $\epsilon(i)$ represents the rounding error. Defining $\mathcal{E}(k) = \{\epsilon(i)\}$ as the set of $i_{\max}$ error samples associated with parent node $\mathbf{v}_d(k)$, the set containing all possible children from $\mathbf{v}_d(k)$ is expressed as

$$\mathcal{V}_u(k) = \texttt{round}(s \cdot \mathbf{v}_d(k)) + \mathcal{E}(k). \tag{3.6}$$

Thus, the geometry from the NNI upsampling is given by

$$V_u = \texttt{unique}\left(\bigcup_{k=1}^{K} \mathcal{V}_u(k)\right). \tag{3.7}$$

The colors for $\mathcal{V}_u(k)$ are defined as $C_u(k) = \{\mathbf{c}_u(i)\},\ i = 1, 2, \ldots, i_{\max}$, such that, $\mathbf{c}_u(i) = \mathbf{c}_d(k)$, and

$$C_u = \bigcup_{k=1}^{K} C_u(k). \tag{3.8}$$

While NNI provides a simple and computationally efficient upsampling strategy, it can introduce aliasing artifacts due to its coarse interpolation. To improve the quality of the upsampled geometry V_u, additional smoothing operations can be applied to refine both the geometry and texture. One commonly used technique is Laplacian Smoothing (LS) [174], originally developed for mesh smoothing but adaptable for point clouds. Briefly, with LS, for each occupied voxel in a given input point cloud, its $3 \times 3 \times 3$ neighborhood is analyzed, and the average voxel position is computed. This average is rounded to the integer grid, and it becomes the center voxel's new position. For attribute smoothing, each voxel inherits attributes from its nearest neighbors. If an exact match exists in the input point cloud, it retains its original attributes; otherwise, an average of the nearest neighbors' attributes is assigned. By applying LS, the aliasing introduced by NNI can be reduced, improving the overall visual and geometric consistency of the upsampled point cloud.

3.3 INTRA-FRAME SUPER-RESOLUTION OF VOXELIZED POINT CLOUDS

The idea of using a dictionary of child nodes for SR was first introduced by Garcia *et al.* [154] in the inter-frame case, where they aimed to super-resolve LR frames that had been downsampled by a scale factor of $s = 2^n$. Their approach involved constructing a look-up table (LUT) that mapped neighborhood configurations to child node occupancies, using a HR reference frame as ground truth. SR was performed by matching LR voxel neighborhoods to the corresponding child occupancy patterns stored in the LUT.

Our method builds upon this dictionary-based approach but introduces some key modifications. Instead of relying on an external HR reference, the dictionary is built using the same point cloud that is to be super-resolved. We introduce fractional downsampling, which enables SR for arbitrary scale factors rather than being restricted to powers of two. A geometry classification step is incorporated to handle the irregularities introduced by fractional downsampling, preserving parent-child relationships during upsampling.

3.3.1 GEOMETRY FRACTIONAL SUPER-RESOLUTION

An overview of the proposed method is illustrated in Figure 3.5, where each gray-boxed step represents a module in the process.

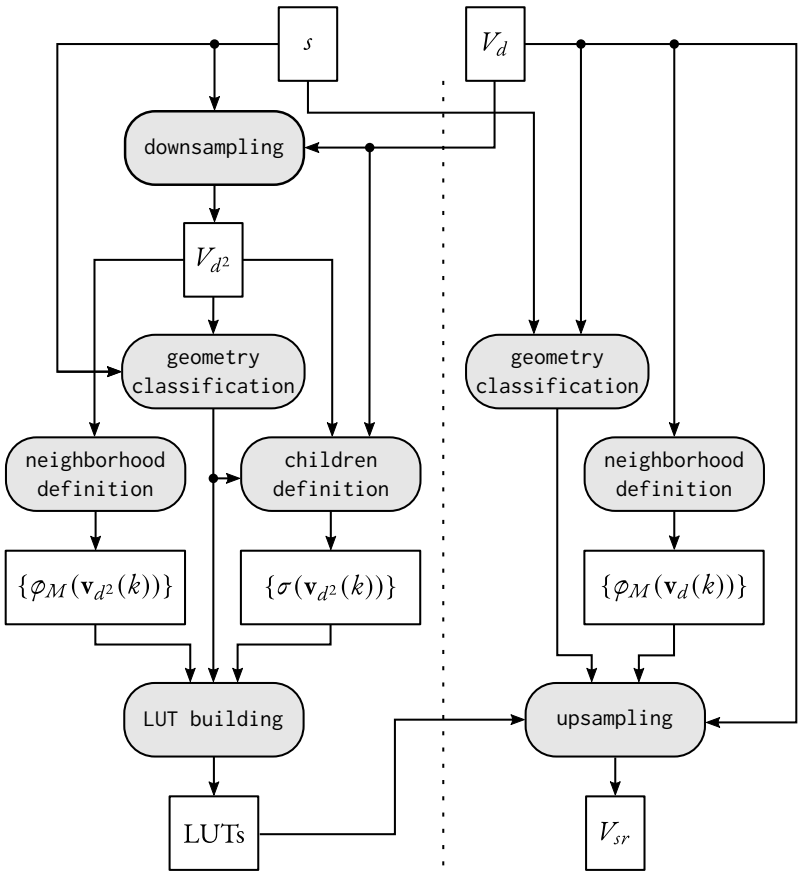


Figure 3.5: General scheme of the proposed SR method.

Neighborhood definition

For each voxel in input LR point cloud V_d, we verify its surroundings to define its neighborhood occupancy. Let $\varphi_M(\mathbf{v}(k))$ be a $(M^3 - 1)$-bit binary descriptor that encodes the occupancy of neighbor voxels inside an $M \times M \times M$ cube around voxel $\mathbf{v}(k)$. The smallest neighborhood, $M = 3$, leads to $3^3 - 1 = 26$ neighbors (adjacent voxels). Thus, a 26-bit descriptor capturing the adjacent voxel occupancy.

Children definition

Let $\sigma(\mathbf{v}_d(k))$ represent the child occupancy configuration of a parent voxel $\mathbf{v}_d(k)$. This is a $\lceil s \rceil^3$-bit binary number, where each bit indicates whether a corresponding child voxel in $\mathcal{V}_u(k)$ is occupied. Differently from [154], we introduce an additional downsampling step on the input V_d to generate a further downsampled V_{d^2} using the same scale factor s.[1] This allows us to define the child occupancy configurations at a coarser level $\sigma(\mathbf{v}_{d^2}(k))$. Figure3.6 illustrates the bitwise mapping between neighborhood and child occupancies.

[1]The assumption of a known s is made so we can compare the super-resolved point cloud with its original version. This is true for compression post-processing applications. When the original value of s is unknown, the method can still be used, but without being able to arrive at the original resolution.

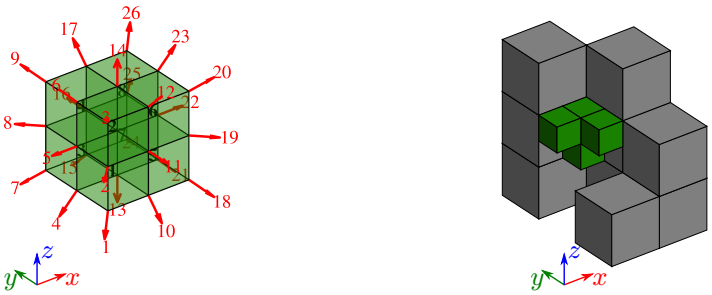


Figure 3.6: On the left, the order of bits representing neighbors (red) and children (black). On the right, an example of neighborhood where φ_3 = `000 111 011 111 00 001 000 000 000` and σ = `1110 1000`.

Geometry classification

To account for irregularities introduced by fractional downsampling, each voxel is classified into a geometry class based on: (i) the number of possible children, and (ii) the positional relationship between uniparous and multiparous coordinates. By assigning voxels to one of eight possible classes (as illustrated in Figure 3.7), we avoid dealing with grid inconsistencies. This module is used whenever we need to estimate child nodes in a fractional grid. It is used when defining child occupancies, during LUT construction (allowing a separate LUT for each class), and in the upsampling module, for correct SR reconstruction.

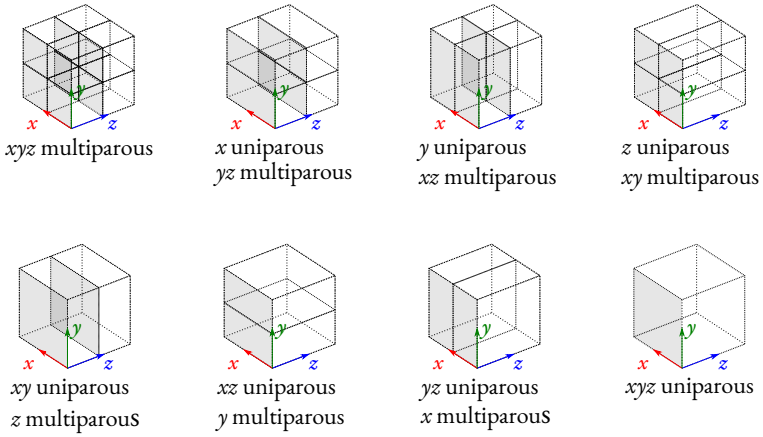


Figure 3.7: The 8 classes of geometry classification for $1 < s \leq 2$.

LUT building

For each voxel in V_{d^2} from each class, we establish a mapping between neighborhood occupancy $\varphi_M(\mathbf{v}_{d^2}(k))$ and child occupancy $\sigma(\mathbf{v}_{d^2}(k))$. This results in a separate LUT for each geometry class, which is used to guide the upsampling process. Each LUT should pair all the 2^{M^3-1} possible neighborhood configurations with an output child occupancy. The m-th entry of each LUT is created by estimating the most likely child occupancy for $\varphi_M(m)$,

$$\bar{\sigma}(m) = E\{\sigma(\mathbf{v}_{d^2}(k)) \mid \varphi_M(m)\}, \tag{3.9}$$

i.e., the expected value (bitwise mean) of all child occupancies sharing the same neighborhood configuration. Neighborhood configurations not present in the input data are associated with fully-occupied child nodes.

To optimize memory usage, only the N_f found neighborhoods are stored, the remaining entries are inferred to have fully-occupied child states in the upsampling module. In this way, each LUT is an N_f-by-2 array, pairing a neighborhood configuration in the first column with its correspondent child occupancy in the second column.

➛ Practical considerations

To ensure a symmetric neighborhood, M must be an odd number. Yet, increasing M significantly impacts computational complexity, as the number of possible neighborhood configurations grows exponentially. For instance, increasing M from 3 to 5 causes the number of configurations to rise from 2^{26} to 2^{124}. As a result, larger values of M become impractical, requiring substantial computational effort for neighborhood searches. Moreover, empirical evaluations revealed that setting $M > 3$ leads to overly specific LUTs, causing the reconstructed geometry to converge toward NNI upsampling, thereby reducing the benefits of the SR method. To balance memory usage and accuracy, we fixed $M = 3$ as the optimal choice. Consequently, whenever $\varphi(k)$ is referenced from now on, it is implicit that a neighborhood size 3, $\varphi_3(k)$, is considered.

Additionally, due to software limitations, our implementation was constrained to $s \leq 3$. Moreover, as s increases, the number of meaningful entries in the dictionary decreases, since there is not much information in the lower levels of the geometry. As s increases, the probability of making a right guess about the estimated children is dramatically reduced, from 1/255 for $1 < s \leq 2$, to 1/19.982 for $2 < s \leq 3$. In other words, the preservation of self-similarities is diminished with the increase of s. Thus, we decided to constrain the values of s, $\{s \in \mathbb{Q} \mid 1 < s \leq 2\}$. Inside this interval, we can profit from partial downsampling and super-resolve a full octree level. If $s > 2$ is required, the proposed SR method can still be used in a cascading manner. For example, by performing $t = \lceil \log_2(s) \rceil$ nested SRs with a new scale factor $s' \approx \sqrt[t]{s}$. Empirically, findings suggest that consecutive upsamplings produce better results than a single one for $s > 2$, despite the increased computational cost.

As a means of data augmentation, we applied incremental translations to the input frame to increase the LUT population. Since $1 < s \leq 2$, shifts of ±1 in each axis are sufficient to change the result of Eq. (3.1), and the parent geometry classification. Other transformations, such as rotation or scaling, are left to future work.

Upsampling

The upsampling module follows a two-step process: (1) NNI is applied top locate all potential child voxels in the upsampled grid; (2) then, the LUTs are consulted to refine the upsampled geometry by removing excess points. Each voxel in V_d is assigned a geometry class and used to query

the appropriate LUT. The LUT determines which child voxels should be removed based on the voxel's neighborhood configuration $\varphi(\mathbf{v}_d(k))$. We super-resolve from V_d to a higher resolution V_{sr} by carving the NNI geometry. The set of super-resolved children from a single parent voxel $\mathbf{v}_d(k)$ is

$$\mathcal{V}_{sr}(k) = \mathcal{V}_u(k \mid \bar{\sigma}(\mathbf{v}_{d,j}(k))), \tag{3.10}$$

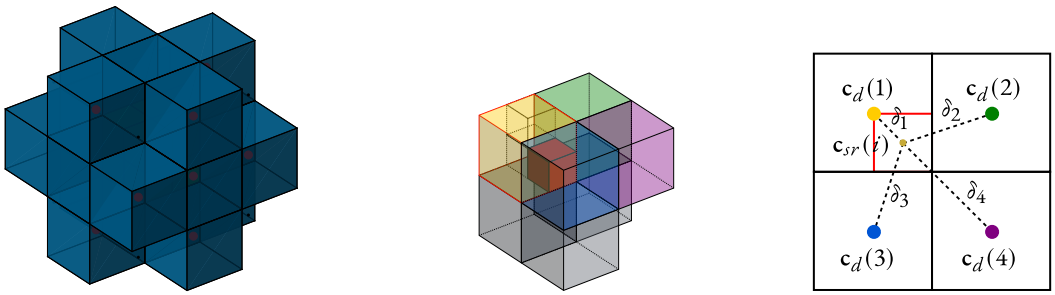
where $\bar{\sigma}(\mathbf{v}_{d,j}(k))$, found by consulting LUT_j for $\varphi(\mathbf{v}_{d,j}(k))$, indicates that a given neighborhood configuration determines which of the possible children of $\mathbf{v}_d(k)$ should be set as occupied, and $0 \leq j \leq 7$ indicates the geometry class (Figure 3.7). The super-resolved geometry is the union of all $\mathcal{V}_{sr}(k)$ as

$$V_{sr} = \bigcup_{k=1}^{K} \mathcal{V}_{sr}(k). \tag{3.11}$$

3.3.2 Color interpolation

To determine the texture of the SR geometry, a common approach is to first expand the LR geometry to match the scale of the SR point cloud. Then, the color for each SR voxel is interpolated using a distance-based weighted average of the LR colors, typically considering a $3 \times 3 \times 3$ neighborhood.

However, a more accurate color interpolation method can be derived by adapting the color prediction scheme used in G-PCC. In the transform domain prediction of RAHT coefficients [10], [90], the estimated color for each occupied child is the average of the parent's color, with the colors of the "uncles" that share an edge with that child. This interpolation strategy is illustrated in Figure 3.8. The weights are determined based on the inverse distance between each parent and uncle relative to the child being estimated. We refer to this modified interpolation method as the weighted average of adjacent neighbors (WAAN). Additionally, we introduce a scaling-dependent weight factor ζ, which adjusts the relative influence of the parent's color as the scale factor s varies. As s decreases, the parent's color should carry more weight in the interpolation.



(a) (b) (c)

Figure 3.8: Illustration of the neighbors used in the WAAN calculation. In (a), all the neighbors that must be considered in the $3 \times 3 \times 3$ are shown. In (b), only the uncle nodes sharing a face with the highlighted child node are depicted. In (c), the illustration of how the distances are calculated. Only uncle nodes from the top were shown to ease representation, but the three uncle nodes from the bottom should also be considered in the calculation.

Let $C_{sr}(k) = \{\mathbf{c}_{sr}(i)\}$ denote the set of interpolated colors for the super-resolved voxels $\mathcal{V}_{sr}(k)$ associated with a given parent voxel $\mathbf{v}_d(k)$. The estimated color for each super-resolved voxel is computed as

$$\mathbf{c}_{sr}(i) = \frac{\mathbf{c}_d(k) + \zeta \sum_{\ell} \delta_{\ell}^{-1} \mathbf{c}_d(\ell)}{1 + \zeta \sum_{\ell} \delta_{\ell}^{-1}}, \tag{3.12}$$

where ℓ is the index of the occupied voxels in V_d sharing an edge with $\mathbf{v}_{sr}(i)$, and the δ_ℓ are their Euclidean distance (Figure 3.8(b)). ζ was empirically found as

$$\zeta = \delta_1 s/8. \tag{3.13}$$

The final set of super-resolved colors is then computed as

$$C_{sr} = \bigcup_{k=1}^{K} C_{sr}(k). \tag{3.14}$$

3.4 PERFORMANCE ASSESSMENT AND ANALYSIS

3.4.1 DATASETS AND TEST CONDITIONS

(a) *longdress* (b) *boxer* (c) *thaidancer* (d) *basketball_player* (e) *queen* (f) *ricardo9*

(g) *head* (h) *statue_klimt* (i) *biplane* (j) *arco_valentino* (k) *facade_09* (l) *house*

Figure 3.9: Representative viewpoints of some of the human figures from (a) to (f), and of the objects from (g) to (l).

For the evaluation of the SR method, we focused on point clouds of static objects and scenes sourced from MPEG's G-PCC Common Test Conditions (CTC) [63]. Due to memory constraints in our current implementation, we set a point cap of approximately 4 million voxels. Table 3.1 provides an overview of the tested point clouds, categorized based on shared geometric properties. We also define the density measure $\rho_{\mathcal{P}}$, representing the average number of occupied neigh-

bors per voxel. To normalize this value, we divide it by 26, which corresponds to the maximum number of adjacent neighbors in a $3 \times 3 \times 3$ neighborhood.

Additionally, the "Vox." column in the table indicates whether a revoxelization pre-processing step was necessary. Revoxelization was applied in two cases:

1. To reduce the size of point clouds exceeding our memory limit (more than 4M points).
2. To increase the density of extremely sparse point clouds, where a downsample using $s = 2$ would decimate less than 1% of the original points.

Figure 3.9 illustrates the selected point clouds.

The density measure ρ_φ can serve as a performance predictor for the proposed method, as it reflects the extent to which self-similarities are preserved across different scales. These self-similar structures tend to be maintained in dense point clouds but degrade in sparser representations. Through empirical analysis, we observed that when $\rho_\varphi \gtrapprox 0.3$, the point cloud exhibits watertight projections—meaning when considering a one-to-one relationship between rendered pixels and voxels, no holes are present.[2] We consider those point clouds with watertight projections to be relatively dense. Conversely, when $\rho_\varphi \lessapprox 0.3$, the point cloud is likely to present holes in its projections, indicating a sparser structure. In the case when $\rho_\varphi = 0$ for a given neighborhood size M, but $\rho_\varphi > 0$ for a larger neighborhood size $M' > M$, then it is possible to redefine this point cloud as denser at a coarser resolution. In other words, an initially sparse point cloud can be effectively downsampled to a lower resolution, where it becomes denser.

3.4.2 Self-similarities at different scales

The proposed SR method assumes that the geometry of a point cloud exhibits approximate self-similarities at different scales. The dictionaries (LUTs) are constructed using a coarser-resolution geometry and later applied to reconstruct a finer details. To assess the extent of self-similarity across scales, we compare dictionaries generated from the original HR point cloud with those created from the LR version.

To quantify these similarities, we compute the Intersection over Union (IoU), which measures the overlap ratio between neighborhood configurations at different resolutions. Specifically, we evaluate IoU over the set of neighborhood configurations $\{\varphi(\mathbf{v}_d(k))\}$.

Figure 3.10 shows the IoU values for different scale factors. The following observations can be made:

- For $1 < s \leq 1.5$, the similarity remains high, particularly due to the prevalence of uniparous parent nodes, which retain strong self-similar relationships.

[2] These findings consider an orthographic voxel-based rendering approach, with the voxel size adaptively following the grid size for different zoom levels. Other rendering approaches were not tested.

Table 3.1: Summary of tested point clouds.

Point clouds	Vox.	Depth	# voxels	ρ_φ
(a) Group: *8i_vox10* [175]†				
longdress_vox10_1300	✗	10-bit	857,966	0.429
loot_vox10_1200	✗	10-bit	805,285	0.428
redandblack_vox10_1550	✗	10-bit	757,691	0.433
soldier_vox10_0690	✗	10-bit	1,089,091	0.432
(b) Group: *8i_vox12* [176]‡				
boxer_viewdep_vox12	✗	12-bit	3,493,085	0.031
longdress_viewdep_vox12	✗	12-bit	3,096,122	0.027
loot_viewdep_vox12	✗	12-bit	3,017,285	0.029
redandblack_viewdep_vox12	✗	12-bit	2,770,567	0.025
soldier_viewdep_vox12	✗	12-bit	4,001,754	0.026
(c) *Thaidancer_viewdep_vox12* [176]‡	✗	12-bit	3,130,215	0.332
(d) Group: *owlii* [177]‡				
basketball_player_vox11_00000200	✗	11-bit	2,925,514	0.452
dancer_vox11_00000001	✗	11-bit	2,592,758	0.445
(e) *queen_frame_0200*‡	✗	10-bit	1,000,993	0.524
(f) Group: *MVUB* [178]†				
andrew9_0000	✗	9-bit	279,664	0.547
david9_0000	✗	9-bit	330,797	0.542
phil9_0000	✗	9-bit	370,798	0.543
ricardo9_0000	✗	9-bit	214,656	0.550
sarah9_0000	✗	9-bit	302,437	0.538
(g) *Head_00039_vox12*‡	✓	9-bit	938,112	0.532
(h) *1x1_Biplane_Combined_000*†	✓	10-bit	1,181,016	0.567
(i) *Statue_Klimt_vox12*‡	✓	10-bit	483,068	0.209
(j) *Arco_Valentino_Dense_vox12*‡	✗	12-bit	1,481,746	0.025
(k) *Facade_00009_vox20*‡	✓	11-bit	1,560,786	0.165
(l) *House_without_roof_00057_vox12*‡	✓	11-bit	3,638,139	0.247

† https://plenodb.jpeg.org/

‡ https://content.mpeg.expert/data/MPEG-I/Part-05/dataSets/

- For $1.5 < s \leq 2$, the similarity decreases rapidly, corresponding to the transition where multiparous parent nodes become more dominant.
- For $2 < s \leq 3$, the similarity decreases at a slower rate, but low IoU levels indicate reduced effectiveness of the method at large values of s.

Moreover, higher similarity levels are observed in "well-behaved" point clouds, i.e., dense and low-noise, such as groups (a), (c), (d), and (e). Sparse point clouds exhibit lower similarity levels, which tend to decrease more rapidly with increasing s. The exception is group (j), which exhibits a different behavior. However, at $\rho_\varphi = 0.025$ this point cloud is so sparse that most voxels are isolated, in fact each voxel has on average only 0.65 neighboring voxels. As a result, the plotted similarity values are based on a very limited number of entries. It is important to note that although IoU only accounts for exact dictionary matches, a partial match can still be beneficial for our purposes.

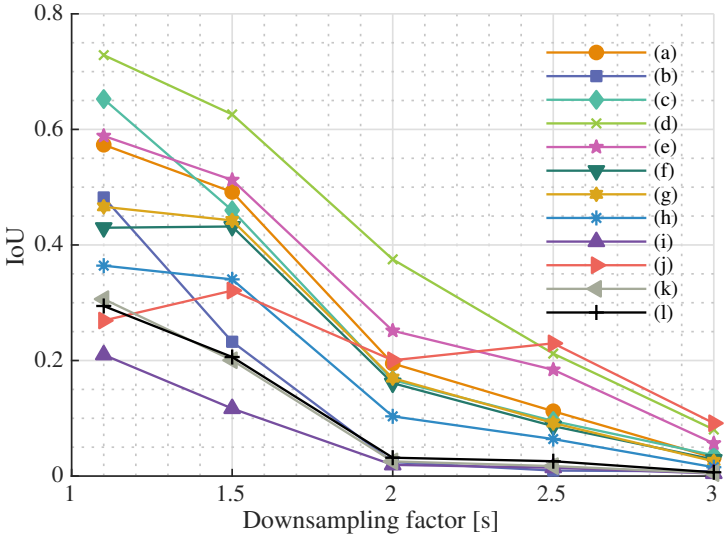


Figure 3.10: IoU measurements comparing LR with HR LUTs at different scale factors.

3.4.3 Evaluation framework

We focused our evaluation in the range of $1 < s \leq 2$, where we believe our method yields best results, regarding complexity and distortion. For $s > 2$, the number of points added by the NNI is quite large, and the self-similarities are diminished, making the NNI carving less effective. We compare the proposed method (LUT) with the NNI, and also considered smoothing it using the LS technique (labeled as NNI+LS). Since the NNI can be replicated by the renderer, just by geometry expansion and using a larger cubic voxel size, we chose not to use the WAAN for its colors to better evaluate the trade-off between complexity and distortion. The alternative for smoother colors is the NNI+LS.

Comparisons with other methods were not carried out because they were developed with set downsampling in mind and would require adaptations to work with grid-downsampled LR point

clouds. Also, most of the deep learning methods would require a lot of retraining to cope with the different scale factors and with the real-world voxelized point clouds of our test set. The methods from Garcia *et al.* [154] also cannot be used because they do not allow for fractional scale factors, nor do they work for intra-frame SR.

3.4.4 Results and analysis

Table 3.2 shows the average performance gains of the NNI+LS and the LUT approaches when compared to the baseline NNI, across all scale factors in the range $1 < s \leq 2$. As observed in the table, the proposed method consistently outperforms both the baseline NNI and its smoothed version (NNI+LS) for nearly all content types and evaluation metrics. The improvements are particularly pronounced when considering point-based geometric distortion metrics, where the LUT-based approach demonstrates superior reconstruction accuracy.

Figures 3.11 and 3.12 further illustrate the geometry distortion comparisons for selected point clouds across different values of s. The results indicate that while the LUT method follows a similar overall trend to NNI, it consistently achieves notably lower distortion levels, reinforcing its effectiveness in preserving geometric fidelity.

Table 3.2: Average performance gains over the NNI for the range $1.1 \leq s \leq 2$.

Point clouds	$D1_{PSNR}$ [dB]		$D2_{PSNR}$ [dB]		Y-PSNR [dB]		PPSNR [dB]		PSSIM		PVIFp	
	NNI+LS	LUT	NNI+LS	LUT	NNI+LS	LUT	NNI+LS	LUT	NNI+LS	LUT	NNI+LS	LUT
(a) *8i_vox10*	3.27	**5.47**	4.29	**5.91**	0.17	**1.95**	2.06	**3.80**	0.02	**0.03**	**0.06**	0.04
(b) *8i_vox12*	-0.63	**1.57**	1.23	**2.73**	-2.79	**2.84**	-1.60	**0.26**	-0.13	**-0.02**	-0.16	**0.02**
(c) *Thaidancer*	2.85	**4.57**	3.67	**5.26**	0.14	**2.92**	2.42	**4.28**	0.01	**0.02**	**0.06**	0.05
(d) *owlii*	3.39	**6.47**	4.41	**6.77**	0.04	**1.61**	2.37	**3.97**	0.01	**0.02**	0.05	0.05
(e) *queen*	3.18	**6.24**	4.66	**7.12**	-0.99	**0.76**	1.27	**3.84**	0.01	**0.02**	0.05	**0.08**
(f) *MVUB*	2.72	**3.98**	4.10	**4.80**	-0.37	**1.31**	0.12	**1.72**	0.01	**0.02**	0.02	**0.03**
(g) *Head*	2.86	**4.32**	4.23	**5.42**	-0.69	**-0.12**	0.01	**1.43**	0.01	**0.04**	-0.03	-0.03
(h) *Biplane*	2.10	**3.25**	3.41	**4.22**	-0.82	**-0.28**	-0.54	**0.86**	-0.01	**0.01**	-0.05	-0.05
(i) *Statue_Klimt*	**0.88**	0.75	**1.75**	1.16	-0.94	**1.29**	-0.55	**0.30**	-0.02	**0.01**	-0.06	**-0.03**
(j) *Arco_Valentino*	0.00	**0.93**	0.03	**1.02**	-3.59	**0.01**	-0.09	**1.02**	-0.01	**-0.03**	**-0.08**	-0.30
(k) *Facade_00009*	0.61	**1.43**	1.88	**2.29**	-1.61	**1.58**	-0.93	**0.04**	-0.05	**-0.01**	-0.12	**-0.06**
(l) *House*	0.89	**1.64**	2.18	**2.77**	-1.16	**0.74**	-0.79	**-0.04**	-0.02	**-0.01**	-0.09	**-0.04**

As expected from Section 3.4.2, the "well-behaved" point clouds—namely, (a), (c), (d), and (e)—achieved the best performance using the proposed method. However, some point clouds contain inherent artifacts that negatively impact both the NNI+LS and LUT-based approaches. These include occluded voxels with inconsistent colors (*queen*), cropped edges (*MVUB*), and noisy geometry scans, all of which introduce distortions that degrade upsampling accuracy.

The NNI+LS method is particularly sensitive to noise and sparsity in the input geometry. In such cases, and at small scale factors, its averaging mechanism introduces a slight voxel shift, which

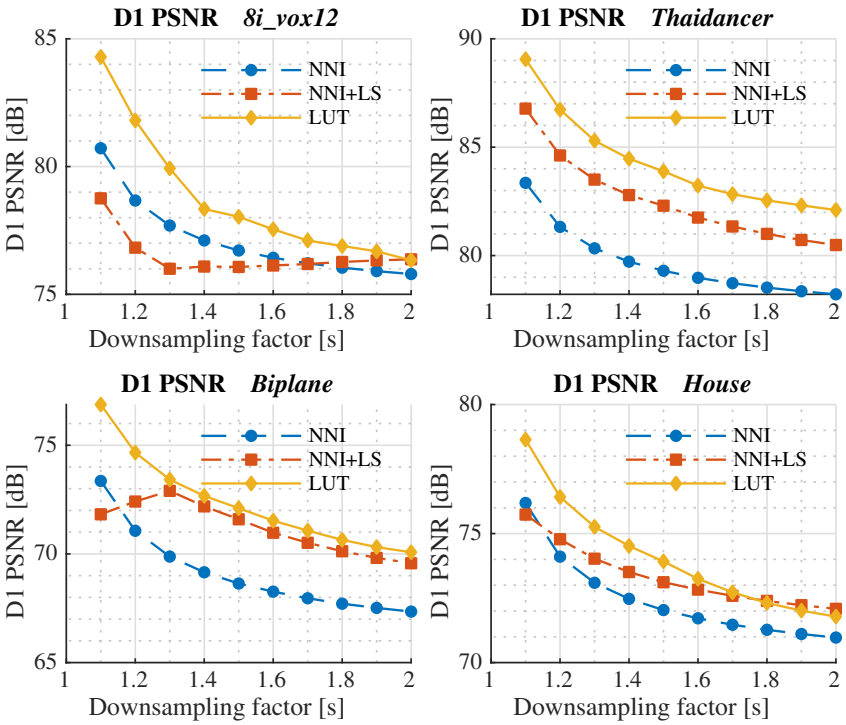


Figure 3.11: $D1_{PSNR}$ metric for point clouds (b), (c), (h) and (l).

degrades distortion metrics. This effect is illustrated in Figures 3.11 and 3.12 and it is accentuated due to the way point-based distortion metrics are computed, as they rely on the maximum rather than the average of two-way distance measurements. As s increases, NNI interpolation becomes more uniform, reducing these voxel shifts and improving overall performance.

Figure 3.13 presents the Y-PSNR plots, demonstrating that the proposed method consistently outperforms the baseline approaches across all tested point clouds, with the exception of Biplane. It is important to note that texture-based metrics tend to be more content-dependent, whereas geometry-based metrics are more influenced by the point cloud acquisition process.

The poor performance in Y-PSNR, for cases like *Biplane*, is primarily due to the high level of noise in its texture, which is difficult to reconstruct using smoothing-based interpolation techniques. It is interesting to notice the under-performance of texture for the NNI+LS method. For lower values of s, many voxels already have the correct color and applying a smoothing filter degrades the overall texture. This is even worse for sparse point clouds, where fewer neighbors are available, which may cause abrupt changes of colors. For $s > 1.5$, texture metrics for NNI and NNI+LS approaches are comparable, but since the latter was worse for smaller s, it ended up being worst overall on average. The main causes for this are noise and the absence of texture information in the neighborhoods for sparse point clouds.

Figures 3.14, 3.15, and 3.16 illustrate the results for projection-based quality metrics. Unlike point-based distortion measures, these metrics are significantly influenced by rendering choices, which affect both projection-based evaluations and subjective assessments. The presence of holes due to missing occupied child voxels has a greater impact on these metrics compared to point-based

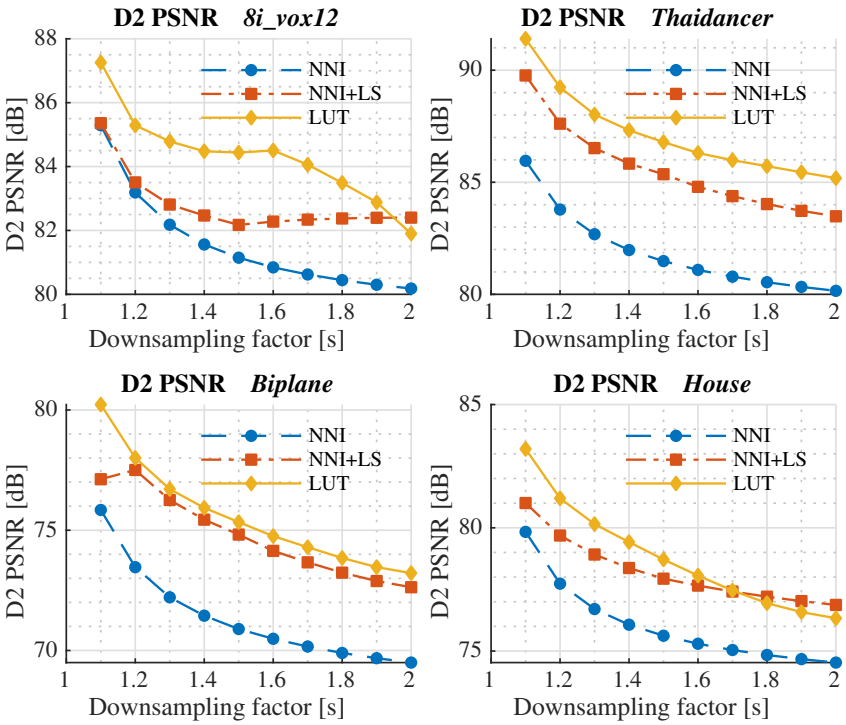


Figure 3.12: $D2_{PSNR}$ metric for point clouds (b), (c), (h) and (l).

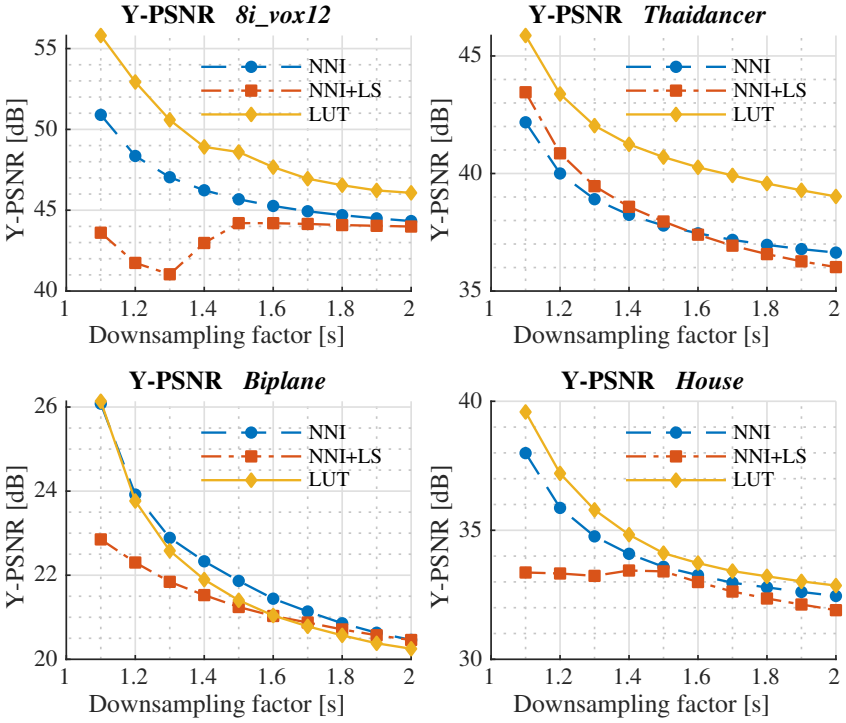


Figure 3.13: Y-PSNR metric for point clouds (b), (c), (h) and (l).

ones, particularly in sparser point clouds, where such artifacts occur more frequently.

PPSNR results share some correlation with point-based metrics since all of those are distance-based metrics, however, the former evaluates the point cloud as a whole differently from the latter,

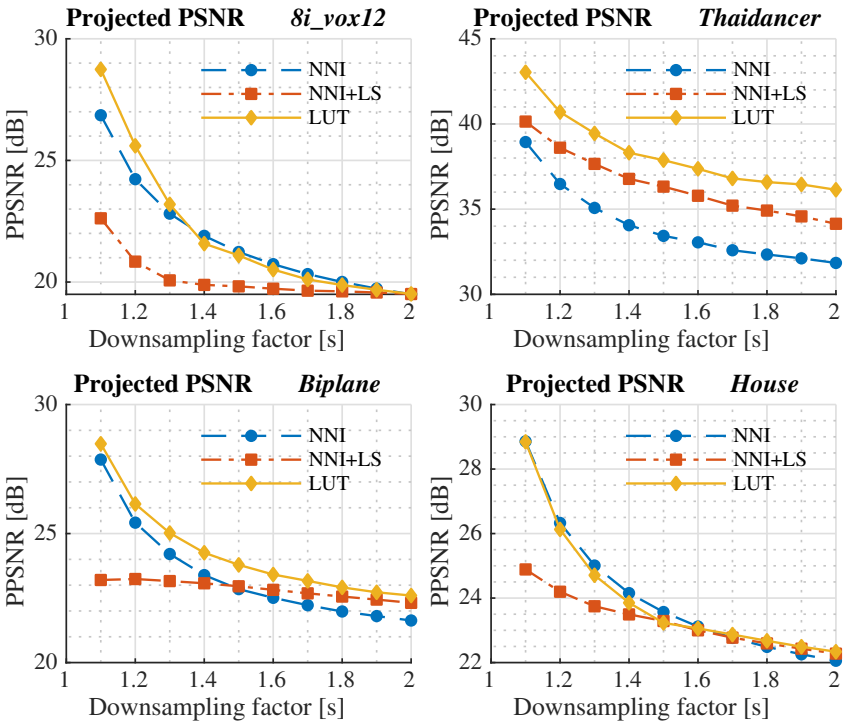


Figure 3.14: Projected PSNR metric for point clouds (b), (c), (h) and (l).

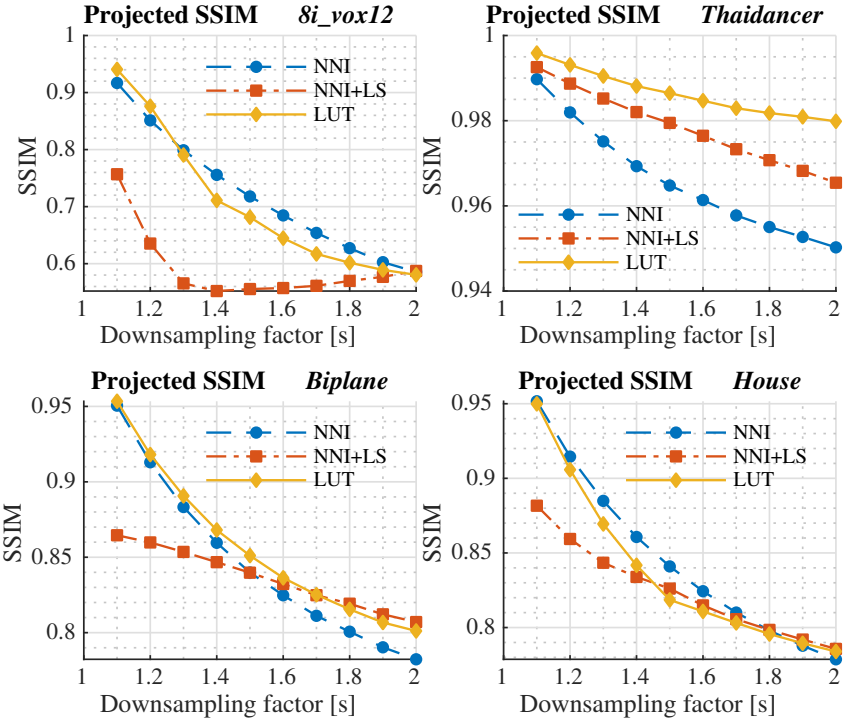


Figure 3.15: Projected SSIM metric for point clouds (b), (c), (h) and (l).

which separately considers geometry and attributes. The low-values of PPSNR observed, particularly for sparse point clouds, could be increased if we changed the splat size, thus making those point clouds look denser, more comparable with the outputs from the upsampling methods. PSSIM and

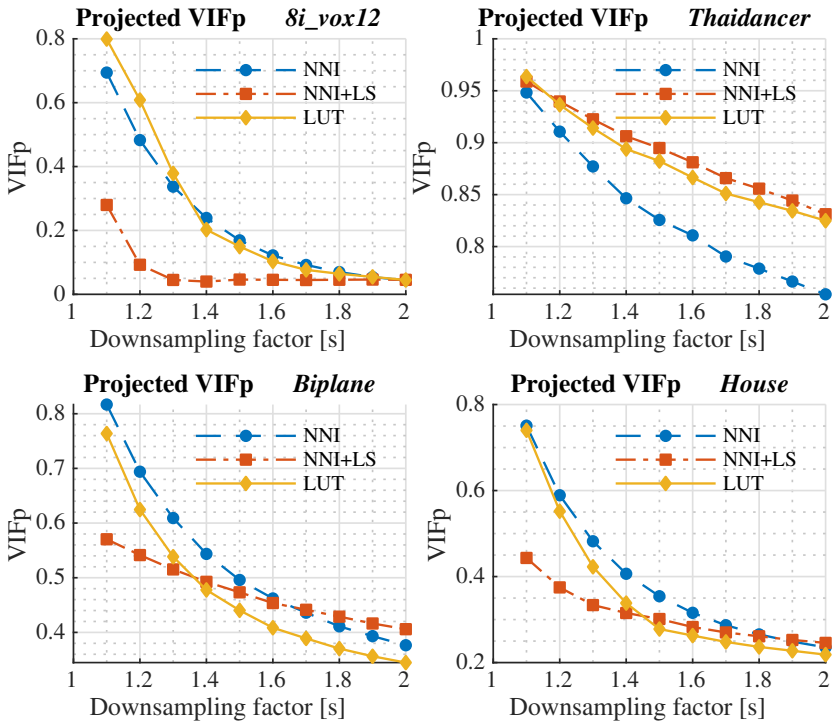


Figure 3.16: Projected VIFp metric for point clouds (b), (c), (h) and (l).

PVIFp are optimized for natural images, so it is hard to analyze their results for the chosen rendering choice, since upsampling and rendering artifacts are mixed. Thus, they cannot capture well the distortions from the outputs of the upsampling methods in sparse point cloud projections. They show a slight preference for the denser clouds from the NNI approach, which resemble more natural images, but are very content-depended, as pointed out in [53]. For a more meaningful understanding of how these metrics behave in the SR context, the rendering approach should be changed and subjective tests performed.

For visual comparison, Figure 3.17 presents viewpoint projections, comparing the ground truth (GT) with the three upsampling methods for *redandblack_vox10_1550* and *Biplane*. These renderings provide additional insight into the perceptual differences between the methods. Qualitatively, the projections from NNI are "bulky" and their texture "blocky", with a steep increase in the number of voxels. There are some improvements for the NNI+LS method, but the geometry is still jagged, and the number of voxels is still large. We can see a large improvement for the LUT method, with much of the aliasing removed, a finer texture, and a much more acceptable number of voxels.

Additional tests were conducted using point clouds (a), (f), (g), and (h) for scale factors s > 2, with evaluations restricted to the $D1_{PSNR}$ metric, as presented in Figure 3.18. Those higher scale factors were achieved using the cascading method LUT approach. Although we can still observe some gains in the geometry for the LUT method over the NNI for $s > 2$, in Figure 3.18, the output point clouds have too many points, and in such cases, it should be evaluated if the added complexity is worth the moderate improvements to a very degraded input geometry. The dips observed at $s = 4$

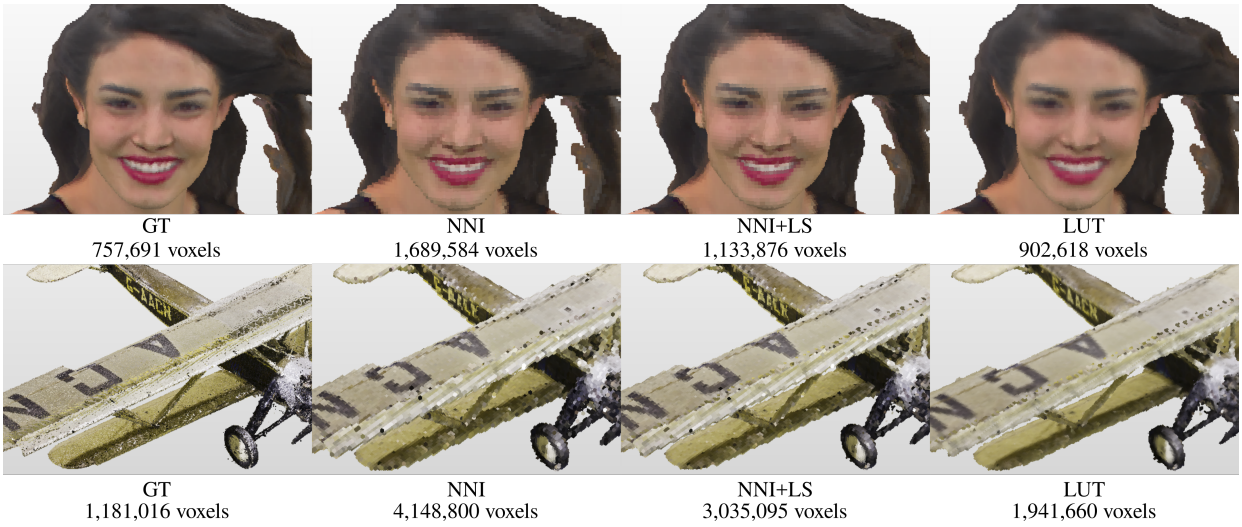


Figure 3.17: Point cloud projections. On the top: *redandblack_vox10_1550* for $s = 2$. On the bottom: *Biplane* for $s = 4$.

and $s = 8$ occur because of the use of consecutive $s = 2$ upsamplings, which is the least effective one for geometry carving. For other values of s, carving is more aggressive, and results are better. This is not ideal and is a point for improvement.

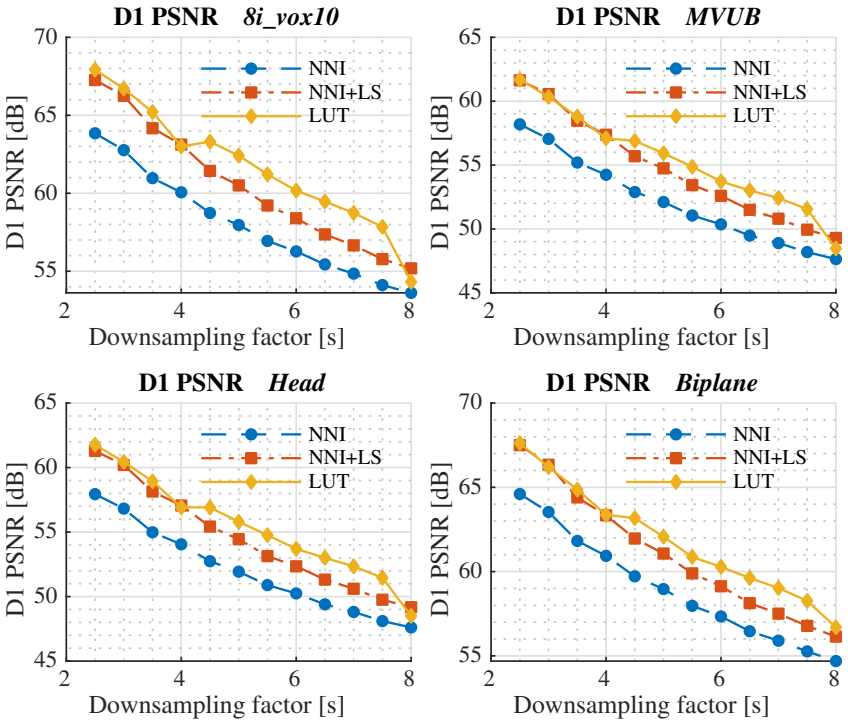


Figure 3.18: D1 metric for point clouds (a), (f), (g) and (h), for $s > 2$.

In Table 3.3, the time profiling for some methods and point clouds are shown. The current implementation was developed using Matlab®/Octave® on a personal computer with Intel Core i7-8550U CPU @ 1.80GHz, with cache size of 8MB and 16GB of RAM. It is possible to speed up the code by removing or reducing the data augmentation step, but with poorer results. For example,

over 90% of complexity reduction can be achieved with complete removal of data augmentation, with an average penalty of 2dB on $D1_{PSNR}$.

Table 3.3: Time profiling for the tested methods.

Point Cloud	s	Time [s]				Output points		
		NNI	LS	LUT	WAAN	NNI	NNI+LS	LUT
Longdress 857,966	1.5	0.5	95.6	60.0	3.6	1,556,255	1,032,233	929,587
	2.5	0.7	75.1	128.9	7.6	2,523,083	1,564,587	1,096,984
	5	1.4	171.3	129.6	10.3	5,016,000	3,588,487	1,595,573
Andrew 279,664	1.5	0.2	11.2	17.6	1.1	456,393	302,379	303,353
	2.5	0.2	17.9	35.0	2.2	699,850	461,764	370,598
	5	0.3	83.3	35.3	3.2	1,316,050	957,647	562,290
Head 938,112	1.5	0.9	89.6	63.5	4.3	1,588,662	1,037,831	1,012,341
	2.5	0.7	78.9	131.1	8.3	2,509,250	1,647,976	1,179,319
	5	3.3	168.0	134.3	11.4	4,893,050	3,566,336	1,747,442
Biplane 1,181,016	1.5	0.6	58.0	76.8	6.0	1,898,394	1,294,955	1,316,698
	2.5	0.8	91.8	167.1	11.2	2,845,701	1,916,829	1,541,756
	5	3.4	193.1	147.3	14.0	5,004,325	3,867,418	2,193,157

3.5 USING FRACTIONAL SUPER-RESOLUTION TO IMPROVE LOSSY COMPRESSION OF POINT CLOUD GEOMETRY

Lossy geometry in G-PCC can be achieved either by just using a pruned octree or, when the *trisoup* method is enabled, by building a surface interpolation on top of the pruned octree. When using only the pruned octree, the quality of the encoded geometry is tuned by a downsampling, similar to what was presented in Section 3.2.1. The downsampling is performed together with a coordinate transformation at the encoder, such that, for the n-th point of V_d

$$\mathbf{v}_{d_n} = \texttt{round}\left(\frac{\mathbf{v}_n - T}{s}\right), \tag{3.15}$$

where,

$$T = (\min x_n, \min y_n, \min z_n), \tag{3.16}$$

and $s > 1$ is the scale factor [88].

At the decoder side, the scaled geometry is simply expanded and shifted back to its original position using the same values of s and T. Since only expansion is performed at the decoder, i.e.,

$$\mathbf{v}_{dec_n} = \texttt{round}(\mathbf{v}_{d_n} \cdot s) + T, \tag{3.17}$$

the downsampled voxels become quickly sparse with the increasing of s.

Some techniques have been proposed in order to improve coding efficiency of the geometry for the lossy case, such as the use of slicing interpolation [179] inside G-PCC or even completely new approaches, such as using dyadic decomposition [180]. We argue that we can use the proposed SR technique as a post-processing tool for point clouds encoded/decoded with G-PCC, in some sort of interpolative compression. We evaluate the performance of the method and compare the results with some learning-based techniques for PCC. This is done as a way to show that the proposed method is competitive with state-of-the-art methods, while being simpler and more easily adaptable for any given rate.

3.5.1 Post-Processing Lossy Geometry

Since the presented fractional SR can be employed for any arbitrary scale factors, it can be used to improve quality of point clouds downsampled using (3.15), provided that s is known. Thus, it can be used to post-process decoded outputs from the G-PCC codec in lossy-geometry configuration.

According to the findings from Section 3.4:

- better results are found when $1 < s \leq 2$;
- for $s > 2$, the method can be applied consecutively;
- for sparse point clouds, a prior downsampling can make the cloud denser, without removing too many points.

With that in mind, we propose to post-process decoded point clouds from G-PCC, following the diagram in Figure 3.19. The scale factor s' represents the downsampling performed prior to the encoding and it is greater than 1 only for sparse point clouds.

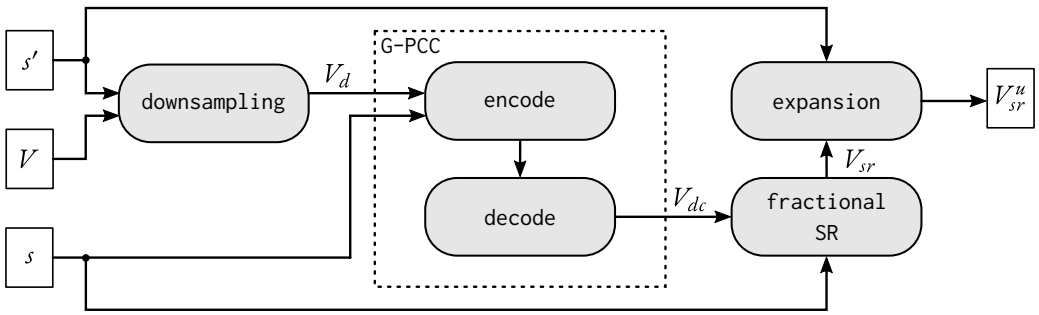


Figure 3.19: Fractional SR applied to decoded point clouds from G-PCC. For sparse clouds, an extra downsampling is applied before G-PCC encoding, and then an up-scaling is applied after the SR process.

3.5.2 Experimental Results

To test the post-processing method, we tuned the value of s' according to the sparsity of the input point clouds:

- for solid (voxelized point clouds with continuous surface), no prior downsampling is performed, $s' = 1$;
- for non-solid point clouds, a prior downsampling was applied, with s' chosen as the largest power of 2 that ensures V_d maintains approximately the same number of points as V.

The definition of solid and non-solid was taken from [181].

This approach selectively downsamples the input geometry only while necessary to increase density, thereby preventing excessive point decimation. According to the CTC's lossy-geometry section, the largest possible value used for this scaling would be 2048 (defined for point clouds with 20 bits in geometry precision). This means we would only need 4 bits to transmit s' as side information. Since the bitrate is calculated in bits per input points, i.e., the number of points in the original point cloud (which vary from hundreds of thousands to millions), these 4 bits are negligible.

The scaling factor, given by

$$s = \frac{1}{\texttt{positionQuantizationScale}}, \tag{3.18}$$

is defined by MPEG G-PCC's Common Test Conditions (CTC) [63] according to the point cloud's geometry for each rate point to be tested, as shown in Table 3.4. The greater the rate id (as in "R6"), the greater the bit rate, the closer to 1 is the value of s.

Table 3.4: The values of `positionQuantizationScale` as used in G-PCC's CTC [63] for each rate point.

Geometry precision	R6	R5	R4	R3	R2	R1
10	15/16	7/8	3/4	1/2	1/4	1/8
11	7/8	3/4	1/2	1/4	1/8	1/16

Table 3.5 provides information about the prior downsampling for each tested point cloud.

We compare our results with those of G-PCC and the following machine learning (ML)-based end-to-end compression techniques: PCC-GEO-CNN-v2 [182], PCGCv2 [183], and ADL-PCC [184]. These methods were analyzed as part of a study on AI-based solutions for PCC, conducted by Zaghetto [185].

PCC-GEO-CNN-v2, by Quach *et al.* [182], focuses on lossy compression of static point cloud geometry using deep convolutional neural networks (CNNs). This method improves over a previous version [186] by introducing several enhancements, including a scale hyperprior model for

Table 3.5: Tested point clouds and their correspondent prior downsampling factor.

Point cloud	s'
dancer_vox11_00000001 [177]	1
longdress_vox10_1300 [175]	1
loot_vox10_1200 [175]	1
queen_0200	1
redandblack_vox10_1550	1
soldier_vox10_0690	1
house_without_roof_00057_vox12	2
statue_klimt_vox12	4

entropy coding, deeper transformation networks, a revised balancing weight in the focal loss, optimal thresholding for decoding, and sequential model training to improve efficiency.

PCGCv2, developed by Wang *et al.* [183], introduces a geometry compression framework based on sparse convolutional networks. This approach supports both lossless and lossy compression while also providing scalable coding capabilities. The method represents the point cloud as a sparse tensor and employs spatially sparse CNNs to process voxel data. These sparse CNNs exploit the spatial dependencies between voxels to predict occupancy probabilities, which are then used for entropy coding or for binary classification of voxel occupancy symbols.

ADL-PCC, also by Guarda *et al.* [184], employs a block-wise deep learning compression approach that adapts dynamically to the content characteristics of the point cloud. The method applies different deep learning coding models selectively to individual blocks, optimizing the encoding process for various types of structures within the point cloud. It incorporates an autoencoder to generate latent representations, followed by discretization for entropy coding, while a variational autoencoder (VAE) estimates entropy model parameters. Additionally, a deep-learning model selection strategy evaluates reconstruction quality and rate efficiency, ensuring an optimal balance between compression and distortion. The final point cloud is reconstructed by merging the encoded blocks.

We report results only for the sequences used in the ML-based tests to enable a direct comparison. Some of the point clouds from G-PCC's CTC were used in the training process of certain machine learning models, making them unsuitable for testing and thus excluded from our evaluation.

Figures 3.20 and 3.21 present the rate-distortion curves for a selection of solid and non-solid point clouds, respectively. The rate is expressed in bits per input voxel (bpiv), corresponding to the number of voxels in the original point cloud, while the distortion is measured using D1 PSNR (Section 2.4)

The results demonstrate a significant improvement over G-PCC, particularly at higher bitrates, where the proposed method consistently achieves better rate-distortion performance. Additionally,

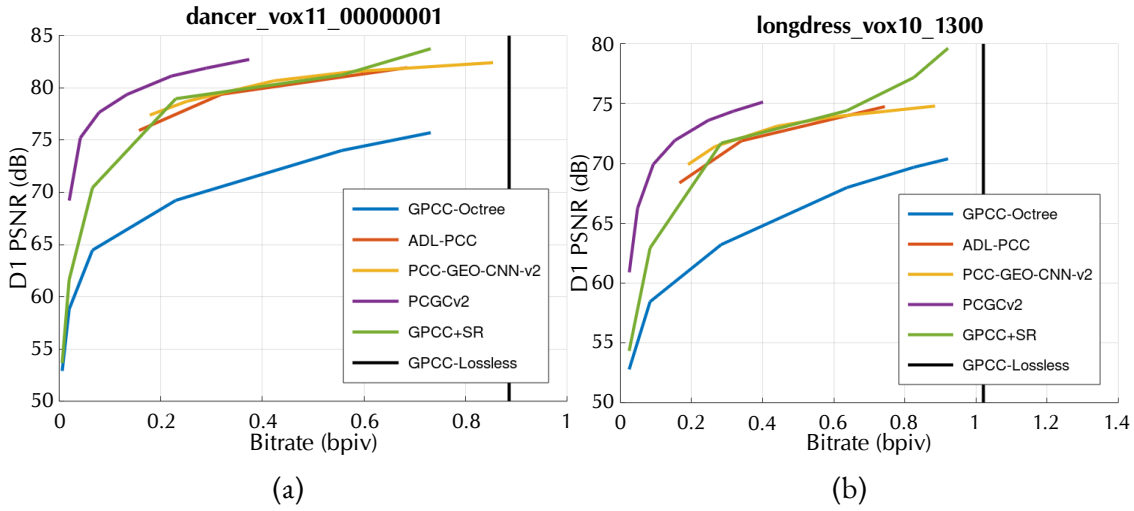


Figure 3.20: D1 PSNR results for solid point clouds: (a) *dancer*, (b) *longdress*.

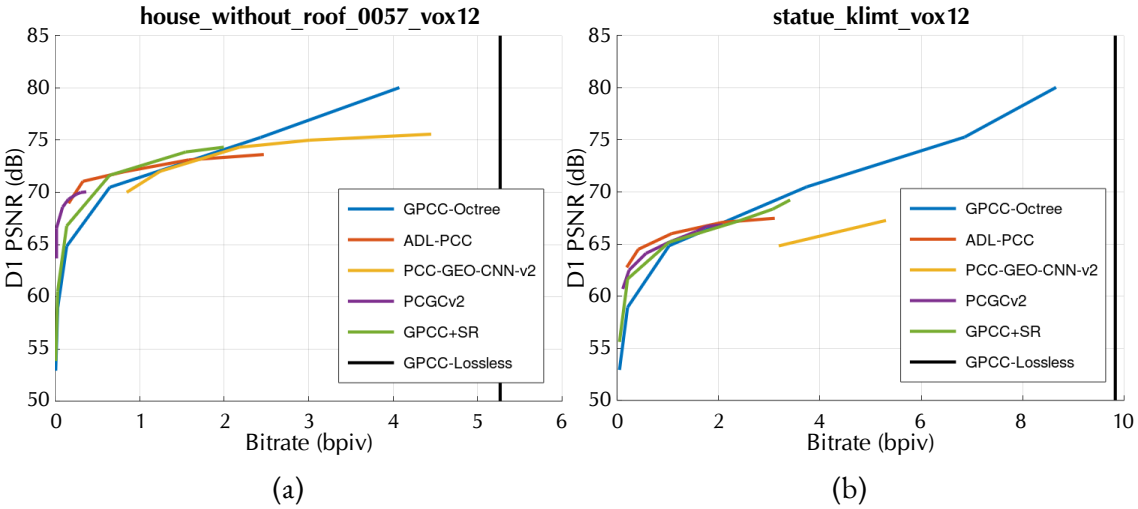


Figure 3.21: D1 PSNR results for non-solid point clouds:(a) *house_without_roof*, (b) *statue_klimt*.

our approach yields comparable results to those of ML-based compression techniques, with the exception of PCGCv2, which achieves superior performance, particularly for solid point clouds.

It is important to highlight that, unlike the proposed method, all ML-based techniques require extensive model training, with separate, fine-tuned models for each target bitrate. In contrast, our approach remains simpler and highly adaptable, capable of efficiently handling arbitrary rate configurations without requiring prior training.

A noticeable performance drop can be observed at lower bitrates. In this region, $s > 2$, necessitating the application of the SR method multiple times. As discussed in Section 3.4.2, this iterative process is not ideal, as each successive SR application modifies the neighborhood structure, which can lead to error propagation across upsampling stages.

Table 3.6 presents the BD-rate comparison of the proposed method against G-PCC, evaluated for both solid and non-solid point clouds. For solid point clouds, an additional column, “frac

Table 3.6: BD-rate comparison against G-PCC for tested point clouds, in rate %. We compare the proposed method to: (A) PCC-GEO-CNN-v2, (B) PCGCv2 and (C) ADL-PCC.

Point cloud	A [182]	B [183]	C [184]	frac SR	frac SR (HR)
dancer	-77.04	-87.81	*-93.63*	-59.79	**-91.77**
longdress	-75.67	-77.06	*-89.09*	-62.39	**-80.30**
loot	-75.86	-74.88	*-90.38*	-64.40	**-81.17**
queen	-75.93	-79.46	*-92.85*	-70.23	**-85.29**
redandblack	-74.11	**-75.88**	*-88.62*	-62.52	-73.89
soldier	-76.99	-75.30	*-89.39*	-65.63	**-80.66**
house	**-41.14**	26.91	*-87.77*	-30.84	-
statue_klimt	*-41.16*	207.18	**-39.50**	-37.80	-

(a) (b) (c)

Figure 3.22: Zoomed in projections of *longdress*. The point cloud was compressed with G-PCC using $s = 2$. We compare different approaches to render the decoded point cloud. In (a) the output from G-PCC ($V_{dc} \times s$), in the original voxel resolution; in (b) the voxel size was increased (same effect as NNI) and in (c) the geometry was super-resolved (V_{sr}). To better highlight geometry differences, texture was kept lossless.

SR (HR)", reports results considering only the four highest rate points, corresponding to the four rightmost points in Figure 3.20. This selection provides a fairer comparison with ML-based techniques, as the rate ranges in this region are closer to those used in ML evaluations.

The results indicate that non-solid point clouds (Figure 3.21) are generally more challenging to compress and super-resolve. Part of this is due to the sparsity of the data, which makes it hard for the ML methods to converge, and decreases self-similarities at different scales. Furthermore, these non-solid point clouds are notoriously noisy, which further degrades compression efficiency and reconstruction quality.

The proposed method's complexity arises primarily from neighborhood search, whereas ML methods typically demand GPU resources. Complexity comparisons among them present challenges as the order of magnitude of operations are quite different and are executed in different architectures, yet we expect the proposed method to be simpler and potentially much faster execution when compared to ML approaches.

(a) (b) (c)

Figure 3.23: Zoomed out projections of *longdress*. Once more, the point cloud is compressed with G-PCC using $s = 2$. This time, the original uncompressed geometry V_{orig} in (a) is compared against the G-PCC output V_{dc} in (b), and with its super-resolved version V_{sr} in (c). To better highlight geometry differences, texture was kept lossless.

Figure 3.22 presents zoomed in projections comparing different solutions to render the decoded geometry from G-PCC, i.e., from (3.15), using a viewpoint from *longdress* for $s = 2$. In Figure 3.22(a), nothing is done, just the the expansion (performed inside G-PCC) to bring the decoded point cloud back to its original size. There we can observe that this causes the points to be sparse, loosing the watertightness of the original geometry. In Figure 3.22(b), we doubled the voxel size comparing to (a) to fill in the gaps. This is similar to the NNI approach, but done by the renderer without changing the geometry itself. We can see that the watertightness is recovered, but details were lost and the geometry is jagged. Figure 3.22(c) shows the result with the post-processing SR (V_{sr}). Here the voxel size is the same as in (a), and the refinement of the geometry is clearly visible.

Finally Figure 3.23 depicts a comparison between the original geometry, the decoded geometry from G-PCC (V_{dc})and the post-processed version (V_{sr}). Again it is clear the improvement in the geometry, with the post-processed version being much closer to the original.

Notice that the colors shown in Figures 3.22 and 3.23 are obtained with a recoloring tool (i.e. lossless colors), for better observation of geometry artifacts.

3.6 CONCLUSIONS

In this chapter we presented a SR method for voxelized point clouds at fractional scales, primarily targeted scales in the range $1 < s \leq 2$. The method leverages self-similarities at lower scales to determine the occupancy of child voxels. Extensive experimental results demonstrate that the proposed method consistently yields lower distortion results when compared to upsampling by NNI, with or without smoothing, particularly when looking at point-based distortion metrics.

For projection-based quality metrics, a broader quality assessment should be performed in order

to better investigate if the subjective preference is maintained. Although our experiments focused static point-clouds only, the method is directly extendable to dynamic ones, where the availability of multiple frames would likely enhance LUT creation and further improve reconstruction quality.

We also applied the proposed method as a post-processing step for point clouds encoded and decoded using MPEG's G-PCC codec. Our comparative analysis with G-PCC and ML-based end-to-end compression methods highlights the significant quality improvements achieved by our approach, particularly at higher bitrates (lower scaling factors) for solid point clouds. In most cases, our method performs on par with ML-based techniques. A key advantage of our approach is that it does not require any prior training, unlike ML-based techniques, which rely on pre-trained, content-specific models tuned for each target bitrate. This makes the proposed method a simpler, more adaptable, and computationally efficient alternative for enhancing point cloud quality in practical applications.

4 ZEROTREE CODING OF SUBDIVISION WAVELET COEFFICIENTS IN DYNAMIC TIME-VARYING MESHES

Dynamic time-varying meshes (TVMs) provide a flexible representation for volumetric video, allowing 3D mesh sequences to describe the evolution of geometry and attributes from real-world captures over time. Unlike traditional animated meshes, which maintain a fixed topology throughout the sequence, TVMs can have varying connectivity, vertex count, and attribute distributions across frames, making their compression particularly challenging. Given their large raw data size, effective TVM compression is essential for efficient storage, transmission, and rendering in immersive applications.

As seen in Section 2.5.2, V-DMC represents TVM geometry using a base mesh together with subdivision wavelet coefficients to describe finer details [137], [187]. While the base mesh is compressed using a static mesh encoder, e.g. MPEG Edgebreaker (MEB) [127], the default method for encoding the resulting wavelet coefficients is to quantize and pack them into 2D images, then compress these images using a standard video encoder such as HEVC [31]. Although this approach benefits from mature video coding tools, it does not fully exploit the inherent hierarchical structure of wavelet coefficients, which could lead to more efficient compression and scalability.

In this chapter, we propose an alternative approach for encoding the subdivision wavelet coefficients, using a zerotree coding strategy that operates directly in the native 3D mesh space. Instead of converting wavelet coefficients into 2D images, we organize them into a spatially coherent hierarchical tree, which better preserves their natural multi-resolution structure. By leveraging zerotree principles (i.e.,hierarchical and embedded coding), the method enables a fully progressive reconstruction at the decoder, where both the *resolution scalability* (already present in V-DMC) across different subdivision levels and the *quality scalability* within a resolution level become possible.[1]

4.1 BACKGROUND AND RELATED WORK

4.1.1 Subdivision Wavelets in V-DMC

In V-DMC surface subdivision is used to encode mesh geometry. The input mesh is simplified (decimated) into a base mesh. To recover the original mesh geometry, the base mesh is progressively subdivided and displacement vectors are computed to approximate the subdivided mesh to the original mesh surface. In such a subdivision scheme, new vertices and faces are added to the simplified mesh by inserting a new vertex at the *midpoint* of each existing edge and uniformly split-

[1] *Resolution scalability* is related to the number of vertices and faces in a mesh; *quality scalability* is related to the precision or accuracy of the reconstructed vertex (*x, y, z*) positions *within* a resolution level.

ting each triangular face into four sub-triangles. This subdivision is applied in an iterative manner, to upsample the base mesh to reach approximately the same spatial resolution as the original input mesh. The intermediate resolution (subdivision) levels provide different LoDs for the mesh. The *displacement* vector field is then determined by finding, for every vertex of the final subdivided mesh, the distance to the closest point on the original input mesh surface. Figure 4.1 illustrates this process in 2D.

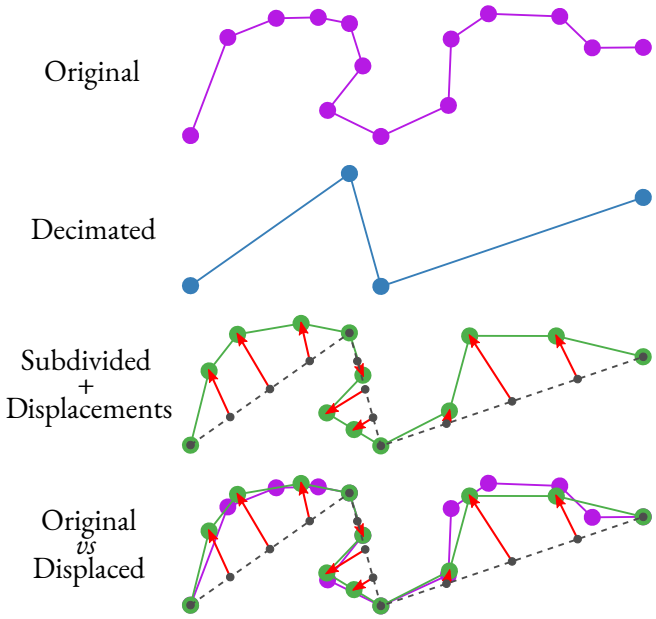


Figure 4.1: Illustration of the computation of displacements in V-DMC. This makes it possible to reconstruct the original mesh geometry in a lossy way, encoding only a small decimated base mesh and a set of displacement vectors, since the subdivision is regular and can be performed at the decoder without any extra connectivity data being sent.

For high-bitrate applications, very little simplification is done to the base mesh, and sometimes displacements are not even encoded. We are more interested in mid- to low-bitrate targets, where the displacement vector is actually used. In this case, the base mesh in V-DMC is still encoded using the MEB, and the displacements are transformed by using the lifting wavelet transform [138], exploiting the subdivision structure. Since the base mesh connectivity can be refined predictably using a set of standard subdivision rules known to both the encoder and decoder, only the base mesh's connectivity information needs to be transmitted, which remains relatively small. Along with the connectivity, the base mesh geometry and a set of encoded wavelet coefficients must also be sent to the decoder. In V-DMC, these wavelet coefficients, together with the base mesh displacements (used solely for surface fitting and left untransformed), are quantized and stored in 2D images, which are then compressed using a standard video codec such as HEVC. Alternatively, the displacements can be encoded using arithmetic coding (AC) [130]. At the decoder, the base mesh—losslessly encoded—is subdivided, and the decoded displacement field is applied to reconstruct the final geometry.

It is important to understand how the *subdivision wavelets* are computed, as this has influenced the choice of coding mechanism that we propose here. In the context of V-DMC, the wavelet

coefficients are computed by applying a lifting wavelet transform to the displacement vectors. This is the simplest form of subdivision wavelets with linear time complexity for analysis and synthesis. In this simple form, the lowest-resolution representation of the mesh geometry is the *base mesh*, i.e. LoD_0, and the wavelet coefficients between any two successive subdivision levels represent the differences between the "child" vertex (x, y, z) positions at LoD_N and the prediction of these child positions using the positions of their "parents" at LoD_{N-1}. For instance, as shown in Figure 4.2, the wavelet coefficient w associated with the displaced vertex v (where v is a higher-resolution level vertex that was originally predicted at the midpoint $\hat{v}$ of a lower-resolution level edge pq) can be computed as:

$$w = v - 0.5(p + q) = v - \hat{v}. \tag{4.1}$$

We see in (4.1) that the resulting wavelet coefficient w is thus the difference between the final (displaced) position of v and its prediction $\hat{v}$. The base mesh can therefore be progressively refined by successive subdivision and addition of more and more wavelet coefficients at higher resolution levels.

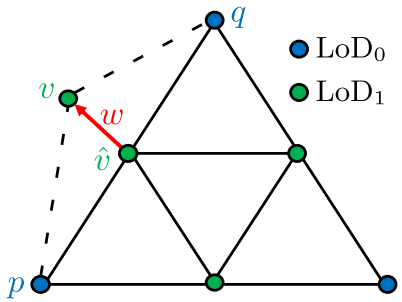


Figure 4.2: Wavelet coefficient computation in linear polyhedral subdivision. The blue vertices are in the base mesh (LoD_0) and the green vertices (LoD_1) are the new vertices inserted at the edge midpoints of the base mesh. The wavelet coefficient w is the difference between the displaced vertex position v and its predicted position $\hat{v}$.

The scheme used in V-DMC for implementing the lifting is shown in Figure 4.3. The displacements (*disp*) are split into "high" (LoD_N) and "low" (LoD_{N-1}) samples. The high-resolution samples H_N are transformed into wavelet coefficients W_N by subtracting the prediction P_N, as in (4.1),[2] and then quantized. The low-resolution samples L_N can optionally be "updated" by adding to them (a fraction of) the sum of the wavelet coefficients of neighboring vertices. The process in Figure 4.3 is repeated until the base mesh (LoD_0) is reached [80]. As mentioned earlier, the decoded displacements must be added back iteratively to the decoded base mesh vertices, to obtain the final vertex positions of the higher-resolution meshes.

4.1.2 Zerotree Coding

The zerotree algorithm is a lossy-to-lossless compression algorithm for a DWT or other hierarchical subband decomposition [145]. It leverages the fact that large low-frequency coefficients of-

[2]Except that, in V-DMC, the lifting wavelet transform is applied on the *displacement vectors* and not on the vertex (x, y, z) positions directly, so the wavelet coefficients represent the prediction errors for the displacements and not for the positions directly.

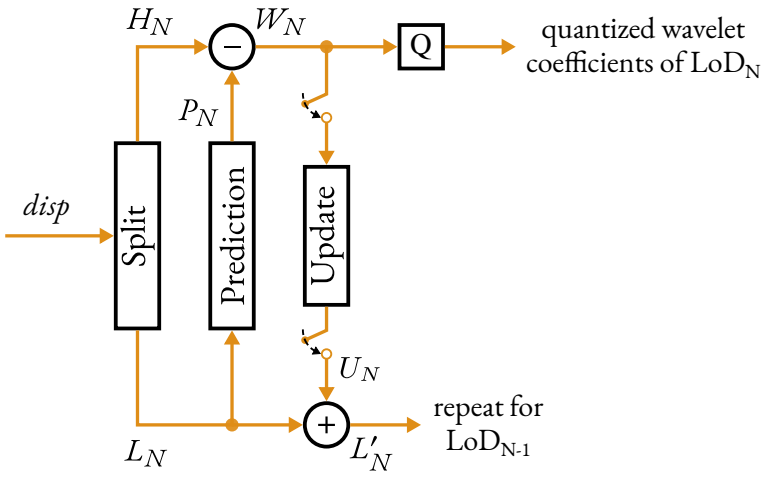


Figure 4.3: Lifting scheme used in V-DMC for computing the wavelet coefficients. The displacements (*disp*) are split into high (current LoD) and low (previous LoD) samples. The high-resolution samples are transformed into wavelet coefficients by subtracting the prediction, as in (4.1), and then quantized. The low-resolution samples are updated using the wavelet coefficients of neighboring vertices. This process starts from the highest LoD and is repeated until reaching the base mesh (LoD_0) [80].

ten indicate significant higher-frequency coefficients, while small low-frequency coefficients tend to imply insignificance in higher frequencies. By structuring wavelet subbands as trees, with low-frequency "parents" and high-frequency "children", the algorithm can compactly encode an entire subtree of insignificant coefficients as a *zerotree*. This term originates from the idea that if a parent coefficient is insignificant, its descendants are likely to be insignificant as well, allowing the entire subtree to be pruned and represented with a single symbol. This approach efficiently represents the significance maps of a hierarchical coefficient structure.

Originally proposed by Shapiro for wavelets on images [145], zerotrees were later adapted to subdivision wavelets on static 3D meshes [188]. In [188], Loop subdivision [136] wavelet coefficients on semi-regular meshes are first organized into an edge-based tree hierarchy and then encoded using a set partition algorithm [146]. The base mesh scaling coefficients are uniformly quantized. In [189], the same SPIHT-based coding system [188] is used to compress unlifted *butterfly* [190] wavelet coefficients of *normal* meshes. In [191], butterfly wavelet coefficients of segments of *normal* meshes are organized into modified versions of the tree structures from [188], in order to be encoded using SPIHT. In [192], the details of butterfly subdivision are similarly encoded using SPIHT. In [193], the authors scale the subdivision wavelet coefficients by applying different weights at different resolution levels, before applying the SPIHT coder.

While these different set partition methods are effective for static meshes, TVMs present additional challenges. Their lack of consistent vertex counts and connectivity across frames requires the computation of a separate tree hierarchy for each frame, thereby limiting inter-frame coding options. The same is true for our proposed method in this chapter.

4.2 PROPOSED APPROACH

In [188], it was observed that when wavelet coefficients are associated with quadrilateral faces in a semi-regular mesh, the hierarchical coefficient structure follows naturally from the face-based subdivision scheme. This leads to a face quadtree, where each parent face subdivides into four child faces, preserving a hierarchical relationship that can be efficiently exploited for compression.

A similar concept applies to *dual* subdivision schemes, where wavelet coefficients are associated with the mesh *faces*. In such cases, the hierarchical structure naturally follows from the face subdivision process: each face at a given resolution level generates a fixed number of child faces in the next level, and the wavelet coefficients describing finer-scale details are directly linked to the evolving face hierarchy. This makes it straightforward to construct a quadtree-like structure for encoding and progressively refining the mesh.

On the other hand, in *primal* subdivision schemes——such as those used in V-DMC——the wavelet coefficients are computed from the mesh vertices and are associated with the edges of the subdivided structure (as illustrated in Figure 4.2). This fundamental difference means that the hierarchical tree must be constructed differently. In the primal case, the vertices of the coarser-level mesh correspond to the *scaling coefficients* of the wavelet transform, while the *wavelet coefficients* are computed at the newly inserted vertices along the edges of the subdivided mesh.

Since each new vertex is generated at the midpoint of an existing edge, the wavelet coefficients are naturally associated with mesh *edges* rather than faces. This results in an *edge-based* tree structure, where each edge at a given resolution level acts as a parent to multiple child edges in the next finer level. Specifically, for a triangular mesh undergoing recursive midpoint subdivision (where each triangle splits into four smaller triangles per level), each edge at a given level spawns four child edges (or fewer if the edge lies on a mesh boundary), all sharing the *same orientation* in the subdivided mesh. This hierarchical edge structure is what we leverage for zerotree-based coding in the proposed method, efficiently encoding the significance of wavelet coefficients across resolution levels. Such a parent-child hierarchy, which we have implemented in our source code, is illustrated in Figure 4.4.

In Figure 4.4, the *root* parent edges (*AB*, *BC*, *AC*, *BD* and *CD*) represent edges of the base mesh, and triangles *ABC* and *BDC* represent two triangles at the base mesh resolution (LoD_0). Each of these triangles is divided into four triangles at LoD_1. In an edge-based tree associated with root edge *BC*, its children are the four sub-edges of the *same orientation* at the next higher resolution level—that is, the sub-edges *BG*, *GC*, *EF* and *HI*. The only wavelet coefficient at LoD_1 that is associated with the base edge *BC* is the one corresponding to the midpoint of that edge (i.e., at vertex *G*). The wavelet coefficients at LoD_2 that are associated with the base edge *BC* are the ones corresponding to the midpoints of its child edges (i.e., at vertices *T*, *P*, *L*, *W*). A similar process can be followed to determine the wavelet coefficients associated with the other base edges of the triangles *ABC* and *BDC*. Note that edge *AB* is a boundary edge, so it has only 3 child edges at the next resolution level (*AE*, *EB* and *FG*); similarly for edge *AC*. For each base mesh edge, we only consider descendant

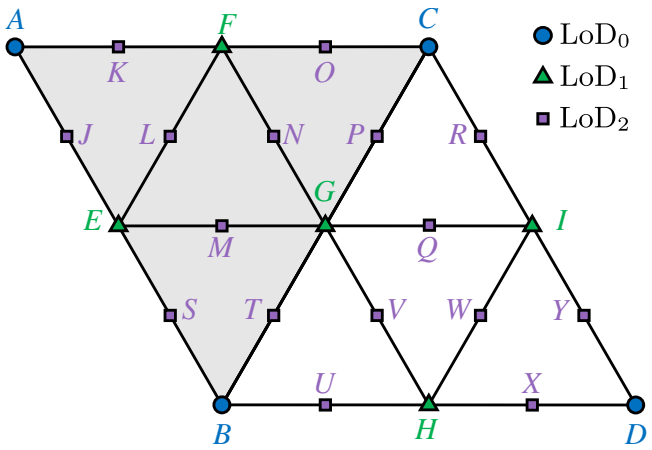


Figure 4.4: Edge-based parent-child relationships for wavelet coefficients in a subdivision surface hierarchy for a triangular mesh.

edges of the *same orientation* as the parent edge, such that no wavelet coefficient is accounted for multiple times or left out.

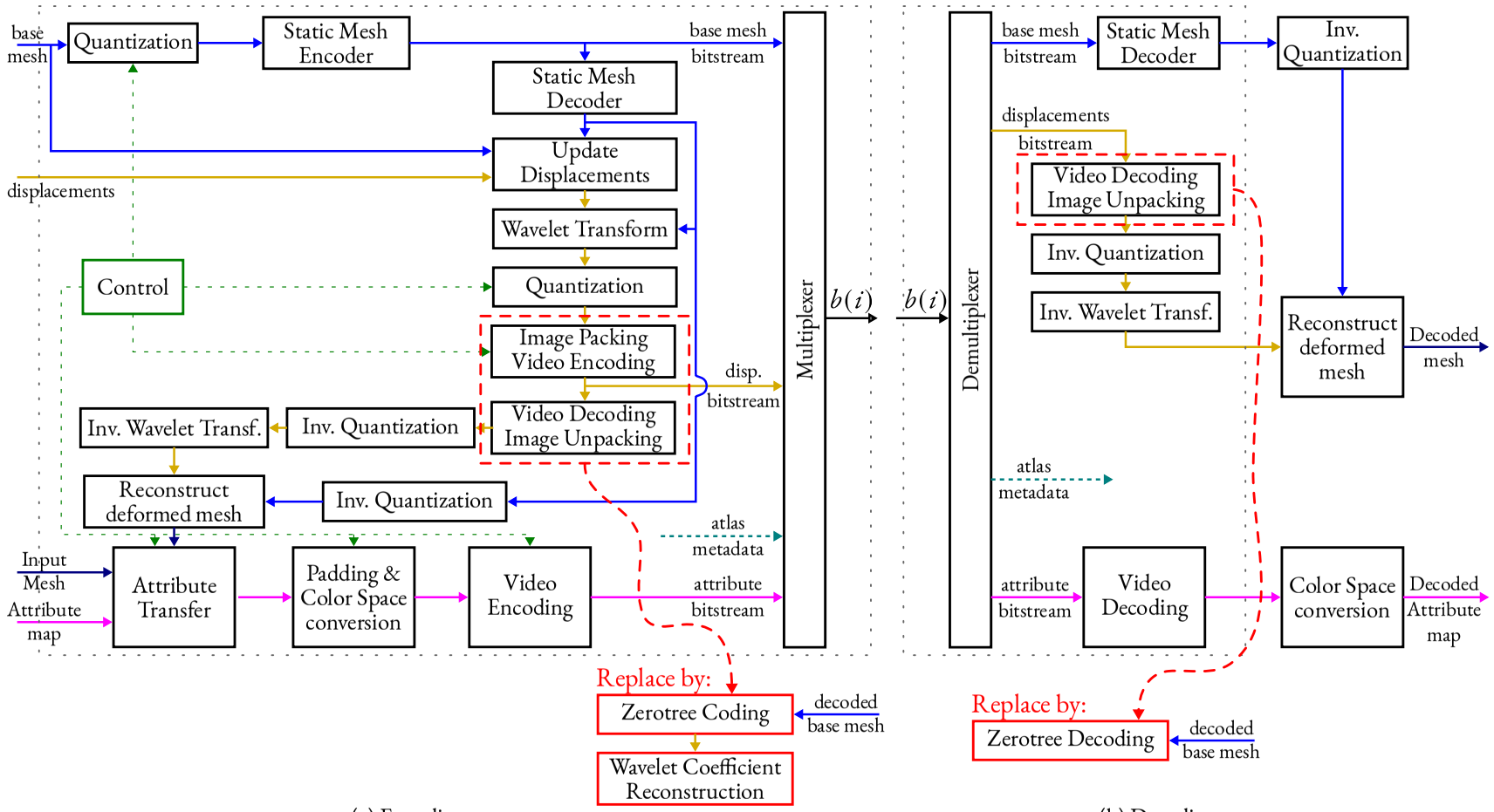
Our proposed modifications to the V-DMC intra-frame encoder and decoder frameworks are shown in Figure 4.5. At the encoder, we replace the **Image Packing/Unpacking and Video Encoding/Decoding** blocks with our **Zerotree Coding** and **Wavelet Coefficient Reconstruction** blocks. At the decoder, the **Video Decoding and Image Unpacking** block is replaced by our **Zerotree Decoding** block. The details of our modifications are explained in the following subsections.

4.2.1 Encoder: Zerotree Encoding Process

The **Zerotree Coding** block from Figure 4.5 is unravelled in Figure 4.6. The encoding method is based on the EZW presented on Section 2.6.1. We assume that the input wavelet coefficients are integers before starting the zerotree coding, which usually means a pre-quantization. This pre-quantization is done in the current V-DMC TM, anyway.

➛ Split Displacements

In the **Split Displacements** block, the base mesh displacements are separated from the wavelet coefficients. The base mesh displacements do not represent wavelet coefficients, but *low-pass scaling coefficients*, as they are associated with the base mesh vertices and not the edges. So, these are not coded with the zerotree coder. They are encoded separately, in our implementation, we encode this data using a Run-Length Golomb-Rice (RLGR) entropy coder.



(a) Encoding process

(b) Decoding process

Figure 4.5: Our proposed changes (shown in red) to the V-DMC intra-frame encoder (a) and decoder (b) [80], [115]. The base mesh (—) is encoded separately using a static mesh encoder. Displacements (—) go through a lifting wavelet transform, exploiting the subdivision structure. Originally, the (transformed) displacements were by default packed into 2D images for video compression; we propose using a 3D zerotree coding approach instead. Input attributes (—) are adapted to the reconstructed mesh and encoded using 2D video coding—we do not change this part.

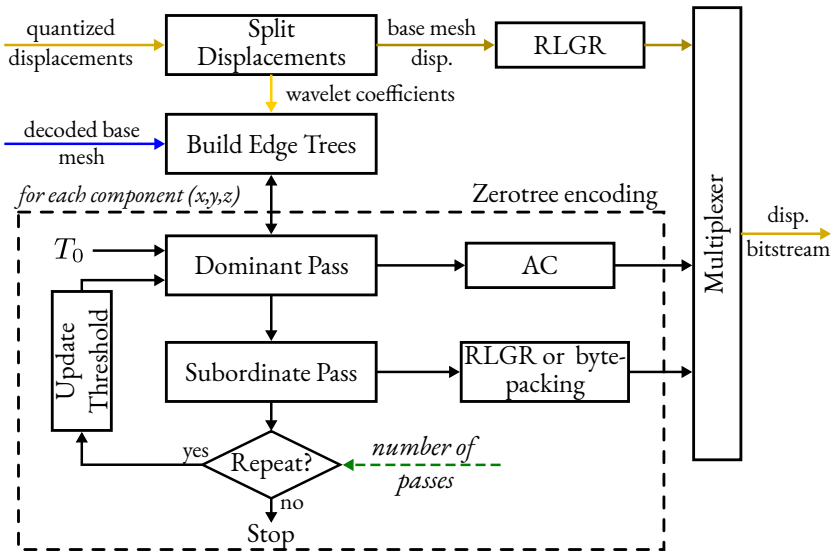


Figure 4.6: Proposed Zerotree encoder.

➛ Build Edge Trees

In the **Build Edge Trees** block, for each unique base mesh edge, we construct an edge-based tree by traversing all the available subdivision levels, as in the hierarchy structure in Figure 4.4, such that the number of trees in the end is equal to the number of unique base mesh edges.

Zerotree coding begins with the choice of an initial significance threshold, T_0, to determine a "significant" wavelet coefficient magnitude. We choose the initial threshold as:

$$T_0 = 2^{\lfloor \log_2(c_{\max}) \rfloor}, \tag{4.2}$$

where $c_{\max}$ is the largest coefficient magnitude in the list.[3]

At each iteration i, the threshold is updated as:

$$T_i = \left\lfloor \frac{T_{i-1}}{2} \right\rfloor. \tag{4.3}$$

We compute a separate threshold for each component (x, y, z)[4] of the wavelet coefficients, and these initial thresholds are transmitted to the decoder. This approach tends to work better than considering a vector quantization method to encode the three components together [188]. This also means that we actually have three zerotrees per unique base mesh edge in our edge-based hierarchy.

[3] In (4.2), we tried replacing $c_{\max}$ by the *average* of the absolute coefficient values, but this performed poorly as the average was near 0 for our datasets, making nearly all the wavelet coefficients significant. Using the *Interquartile Range (IQR) rule* to eliminate high-valued outliers also did not improve results, as these values were essential for reconstructing the higher-frequency details.

[4] The generic 3D component names (x, y, z) are used, but this could also be something else, such as n (normal), t (tangent), b (bitangent), if a different coordinate system is used.

➛ DOMINANT PASS

In the **Dominant Pass**, for each component of the wavelet coefficients, we traverse the edge-based hierarchy in a traversal order that is known to the decoder, and label the coefficients using the binary decision tree shown in Figure 4.7, using the codes:

- ZTR (Zerotree Root): Insignificant coefficient whose descendant coefficients are also all insignificant.
- IZ (Isolated Zero): Insignificant coefficient with one or more significant descendants.
- POS (Positive Significant Coefficient): Significant coefficient whose magnitude is positive.
- NEG (Negative Significant Coefficient): Significant coefficient whose magnitude is negative.

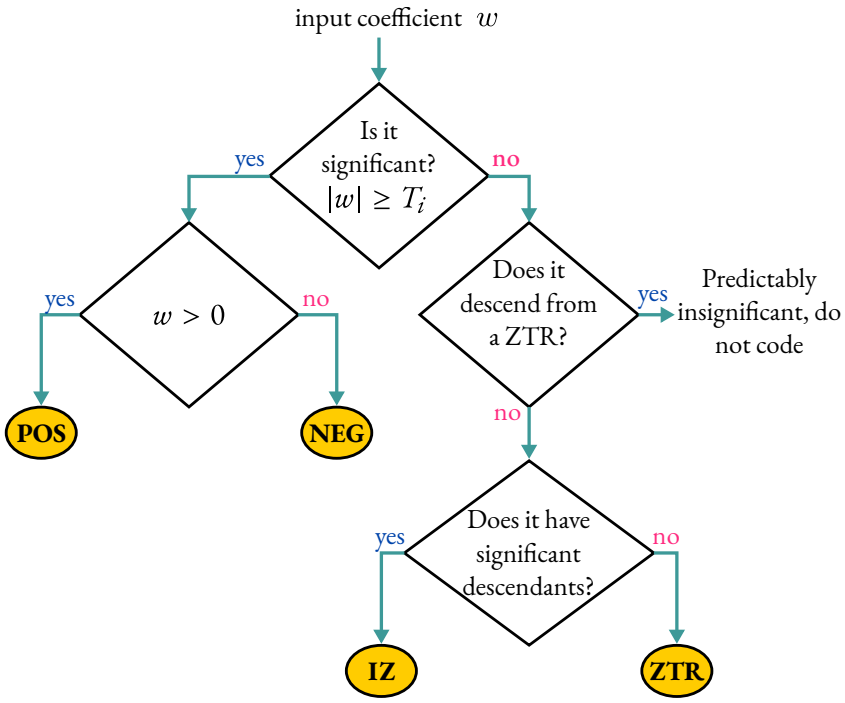


Figure 4.7: Binary decision tree for encoding the significance of a wavelet coefficient using zerotrees.

The chosen scanning order is such that no child node is scanned before its parent node, and all nodes in a given subband (resolution level) are scanned before moving onto the next level. The children of a coefficient are only coded if the parent coefficient is found to be significant, or if the coefficient is an isolated zero. Each of the 4 dominant symbols can be represented using 2 bits, but we choose to entropy-code them using an adaptive arithmetic coder (AC) to produce a lower bitrate. As a result of using the thresholds in (4.2) and (4.3), the magnitudes of significant coefficients in the i-th dominant pass fall within the *uncertainty interval* $[T_i,\ 2T_i)$. These values are then added to a *subordinate list* (separately for each component) for further refinement. Then, we set these significant coefficients to 0 in our tree hierarchy, to exclude them from the next dominant pass.

➛ Subordinate Pass

In the **Subordinate Pass** block, we refine the magnitudes of the significant coefficients by indicating whether they lie in the bottom half of the uncertainty interval, $\left[T_i,\ T_i + \frac{T_i}{2}\right)$, or in the top half, $\left[T_i + \frac{T_i}{2},\ 2T_i\right)$. A `0` bit is output if the value lies in the bottom half, and a `1` bit if it lies in the top half. The order of bits that are output in this refinement pass follows the output order of the significant coefficients from the corresponding dominant pass. If multiple dominant and refinement passes are performed, at each iteration the subordinate list contains significant coefficient magnitudes from previous passes as well, and these are iteratively refined with each new refinement pass, using narrower uncertainty intervals associated with each new threshold. We compare the bitrate of entropy-coding the subordinate pass bits (separately for x, y, z) using RLGR, against directly packing these bits into bytes, and we transmit the smaller output. A binary flag in our bitstream indicates whether or not entropy coding has been applied.

We determine whether to perform additional passes by setting a maximum *number of passes* as a parameter separately for the encoder and decoder (alternatively, this parameter may be encoded into the bitstream). The encoder processes up to this limit, or stops when no further refinement is possible (i.e., when the uncertainty interval length is equal to 1), avoiding the sending of unnecessary bits even if more passes are chosen. The decoder, however, can stop earlier based on its own parameter, allowing different scalable versions of the displacements data to be retrieved from the same bitstream. Alternatively, a target bitrate and/or distortion measure could be used (this is not currently implemented).

Before the next dominant pass, we update the significance threshold (for each component), using (4.3).

➛ Wavelet Coefficient Reconstruction

Finally, in the **Wavelet Coefficient Reconstruction** block in Figure 4.5, the wavelet coefficients are updated with their corresponding reconstruction value, mimicking what happens at the decoder side (see Section 4.2.2), such that texture transfer can be performed on the reconstructed mesh.

4.2.2 Decoder: Zerotree Decoding Process

In Figure 4.8, we show the internals of the **Zerotree Decoding** block from Figure 4.5.

➛ Build Edge Trees

First, in the **Build Edge Trees** block, we reconstruct the edge-based zerotree hierarchy using the decoded base mesh as the starting point, exactly as at the encoder. We then separate the received displacements bitstream into the data corresponding to the encoded base mesh displacements (if

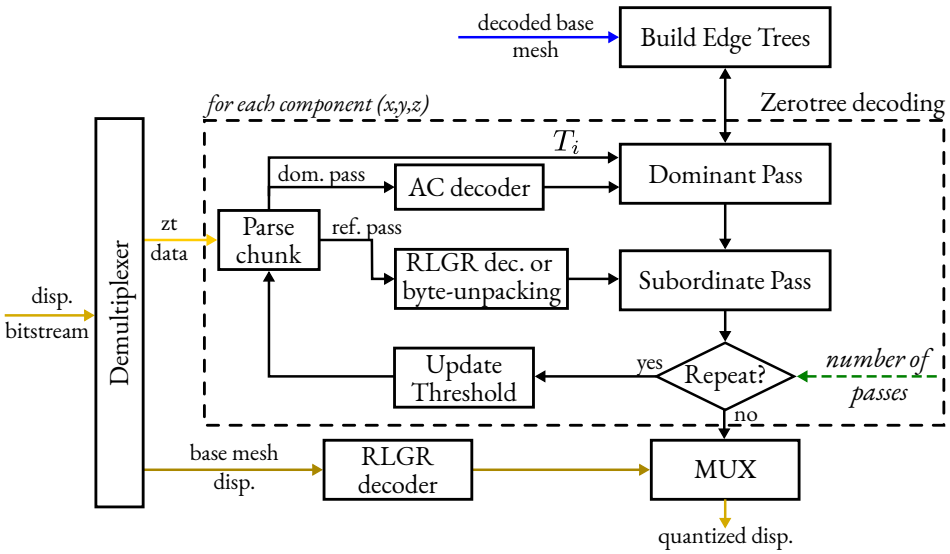


Figure 4.8: Proposed Zerotree decoder.

these exist) and the zerotree-related data (*zt data* in Figure 4.8).

➛ RLGR Decoder

Next, if existent, the base mesh displacements are entropy-decoded (**RLGR decoder** in Figure 4.8). No further processing is needed, since these displacements are encoded losslessly.

➛ Parse Chunk

For the zerotree-related data, in the **Parse chunk** block, we extract data for the current dominant and refinement passes for each component (x, y, z) of the wavelet coefficients, the significance thresholds T_0 for the first dominant pass (if we are at the first pass), and any auxiliary data for entropy-decoding.

➛ AC Decoder and Dominant Pass

We entropy-decode the bits corresponding to the symbols of the first dominant pass in the **AC decoder**. Using the constructed zerotree hierarchy from **Build Edge Trees** and the traversal order that is common to the encoder and decoder, the decoder can replicate the **Dominant Pass** from the encoder using the decoded dominant symbols to obtain the positions of the current significant wavelet coefficients within the zerotree hierarchy. If a given tree node receives a ZTR symbol, the decoder sets the wavelet coefficients for this node, as well as for all of its descendants in the hierarchy, to 0—this is a key aspect of the zerotree coding mechanism, which enables zeroing out insignificant coefficients at the decoder.

➛ RLGR Decoder/Byte-Unpacking and Subordinate Pass

Next, the refinement bits are decoded (using **RLGR dec. or byte-unpacking**, as flagged by the encoder), and we use these to reconstruct the magnitude values of the significant wavelet coefficients decoded so far, in the **Subordinate Pass** block.

➛ Wavelet Coefficient Reconstruction

In order to reconstruct the wavelet coefficient magnitudes, we use the centers of the uncertainty intervals (see Sec. 4.2.1) corresponding to the threshold values, T_i, for the current dominant pass. If a threshold is equal to 1, we choose to reconstruct the corresponding coefficient magnitude as 1 (i.e., the lower limit of the interval $[1, 1.5)$) to obtain an integer value. Similarly, if we reach an uncertainty interval length of 1, we also reconstruct the coefficient as the lower limit of the interval, in order to maintain integer outputs. Note that once we are beyond the first refinement pass, if there exist wavelet coefficient magnitudes to refine from previous passes, for the coefficient reconstruction we first refine the uncertainty interval where the coefficient was found in the previous pass, and then use the center of this refined uncertainty interval.

If there are further dominant passes to decode and/or further refinements to do for the already-decoded significant wavelet coefficients (as indicated by the *number of passes*), we update the significance thresholds using (4.3) and repeat the above steps for the zerotree-related data.

➛ Embedded Bitstream

The decoder can stop the decoding process when satisfied with the reconstructed coefficient values and/or when reaching a desired bitrate limit or a pre-defined number of zerotree passes. Since the received zerotree bitstream is *embedded*, at each decoding of a new set of dominant and refinement pass bits, the signal reconstruction is *complete*, i.e., all the input displacement values have been reconstructed, but the quality (precision) of this reconstruction becomes progressively more accurate as more bits are decoded. Figure 4.9 shows the high-level layout of our embedded bitstream. This bitstream is integrated into the larger V-DMC bitstream by simply replacing the part of the bitstream that was originally associated with *video*-coded displacements. A 2-byte value precedes the zerotree bitstream to indicate its size, which enables the V-DMC decoder to locate it. The zerotree decoder then handles the decoding, parsing, and reconstruction of the displacements data, as explained above. No high-level parameter setting is required by the user, except for choosing a maximum number of zerotree passes to encode/decode (see Sec. 4.2.1).

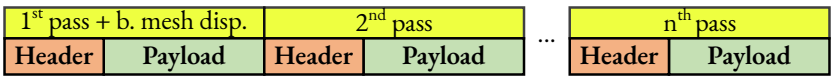


Figure 4.9: Layout of our embedded bitstream, per frame of the input mesh sequence. The first payload also includes the base mesh displacements, while subsequent payloads store only the zerotree data for the wavelet coefficients. The bitstream can be truncated at the start of any header-payload pair to control quality, but the order from the 1st to the nth pass must be preserved.

4.3 RESULTS AND ANALYSIS

In Figure 4.10, we present RD curves for our results on all 300 frames of the V-DMC dynamic mesh dataset [194], without texture coding, using V-DMC TM v4.0 [195], [196]. These RD curves represent the following:

- Lossy intra-coding, i.e., C1 all-intra (AI) configuration.
- HEVC (HM-16.21+SCM-8.8) and AC (schroarith/schroedinger-1.0.11) anchors [195] with the standard v4.0 Common Test Conditions (CTC) rate points and parameters [194].
- A separate RD curve for our zerotree method, for each CTC rate point. Each point on the "zt" curves represents a new zerotree pass, progressively refining wavelet coefficient magnitudes with additional passes.
- x-axes representing the total geometry bits (base mesh + displacements + motion) + metadata and any overhead bits, y-axes representing the geometry D1 PSNR as computed by the CTC metrics [194], across all 300 frames.

To obtain the results in Figure 4.10, we used the V-DMC v4.0 CTC rate points as *targets* for a lossless-displacement reconstruction.[5] Our goal was to prove that, given a starting base mesh and a target subdivision surface (i.e., target rate point and geometry reconstruction quality), our proposed method would enable us to achieve a smooth progression of lossy-to-lossless RD points with an increasing number of zerotree passes. This is evident in our results in Figure 4.10, where we see, for each CTC rate point, a smooth curve of intermediate RD points generated by the zerotree method, each representing a successive dominant and refinement pass. This progression of RD points is produced automatically by our encoder and decoder, with no additional parameter setting by the user.

In Figure 4.11, we compare our zerotree method with the HEVC and AC anchors at the *lossless*-displacement CTC rate points. The results shown here are representative of our results for all 8 input mesh sequences [194]. The zerotree RD curves in Figure 4.11 were obtained by connecting the 5th zerotree pass points from each of the 5 curves for "zt-r0x" (x = given CTC rate point) for the corresponding mesh sequences in Figure 4.10.[6] We can observe in Figure 4.11 that for all the mesh

[5] Lossless with respect to the *pre-quantized* displacement values at that rate.

[6] The number of zerotree passes needed to reach a lossless reconstruction can vary depending on the input distribution of wavelet coefficients.

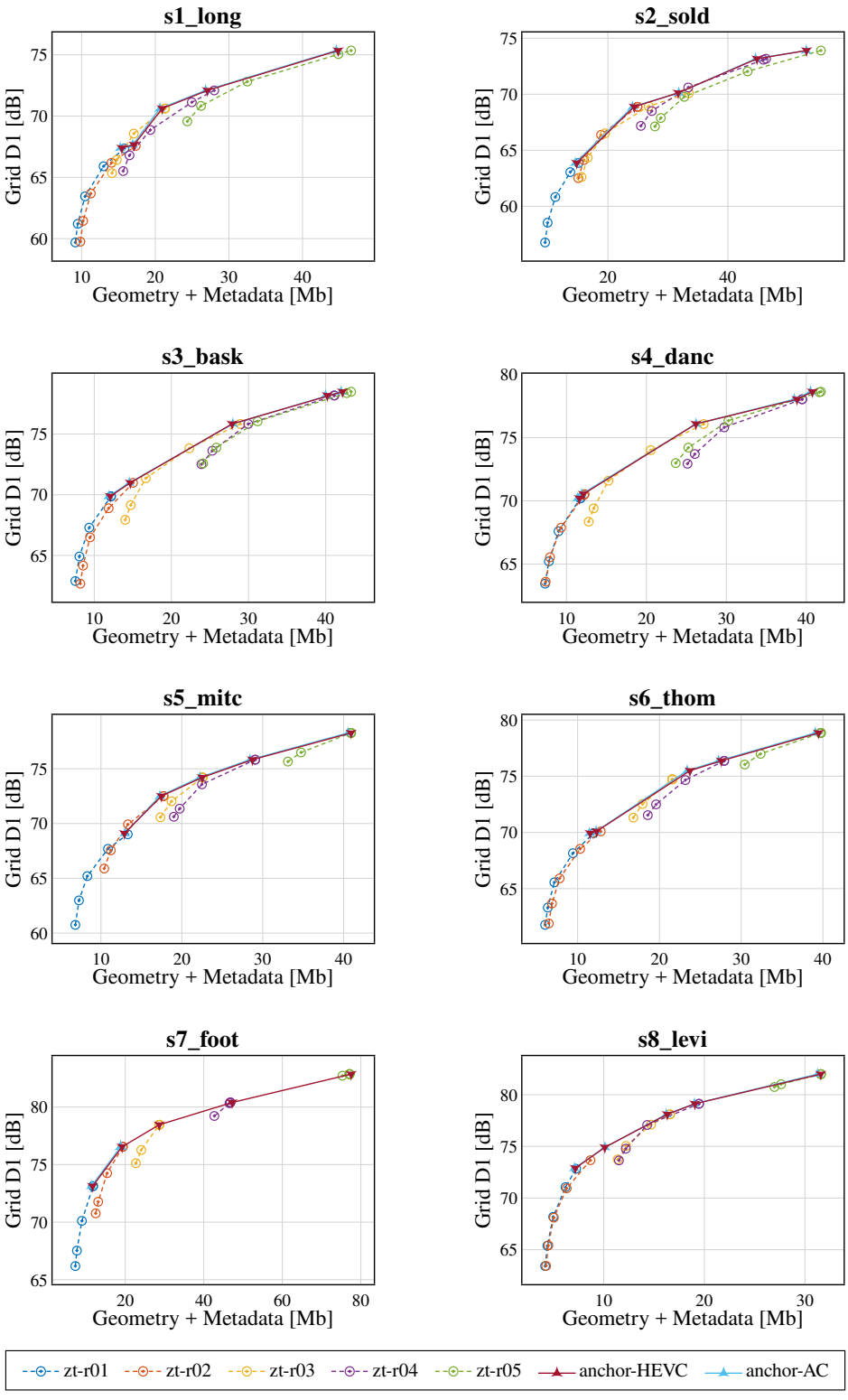


Figure 4.10: RD curves for 300 frames without texture coding, for our integration of the proposed zerotree coding method into the V-DMC TM v4.0.

sequences, our zerotree method is either very close to, or overlapping with, the RD curves of the HEVC and AC anchors. The corresponding BD-rates [50] between our solution and the HEVC anchor are shown in Table 4.1. While the results in Figure 4.11 and Table 4.1 imply that in some cases the zerotree method requires slightly more bits than the anchors for lossless reconstruction, this overhead is justified by the embeddedness and quality scalability of our method, which allows us to easily extract *several quality levels* of progressive geometry reconstructions from the same bitstream, while the anchors encode only one quality level (lossless) at a similar bitrate.

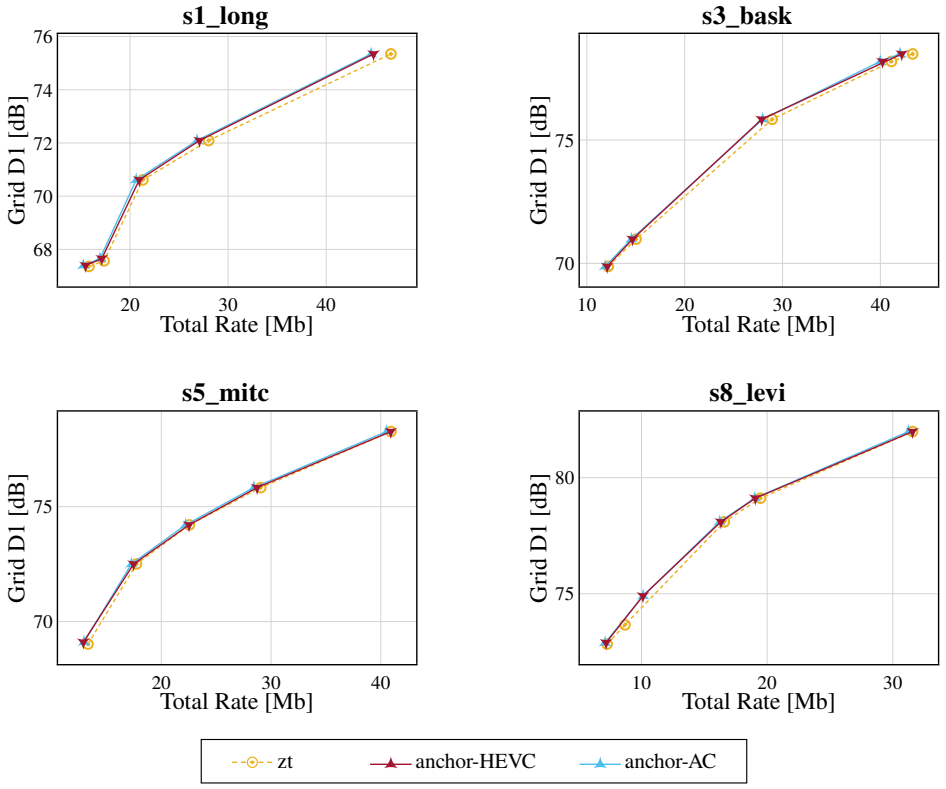


Figure 4.11: RD curves for 300 frames without texture coding, at the lossless-displacement rate points only, for our integration of the proposed zerotree coding method into the V-DMC TM v4.0.

In Figure 4.12, we present some example progressive geometry reconstructions obtained by our zerotree method over multiple dominant and refinement passes, for a given CTC rate point. These reconstructions correspond to the "zt-r01" RD curves in Figure 4.10. Since the proposed zerotree coding is a *quality scalability* approach, not a *resolution scalability* approach, we do not change the number of vertices or triangles across the zerotree passes; we simply refine the positions of the (x, y, z) coordinates of the reconstructed points at a given resolution level. Therefore, in Figure 4.12, the *number of vertices and triangles remains the same across all the zerotree passes*; the only thing that changes is the precision (quality) of the geometry (x, y, z) reconstructions. We show these reconstructions first without, and then with, texture, so that, in the former case, we can more clearly see the finer geometric details that are progressively reconstructed by the zerotree method, which may otherwise be hidden by the texture maps. In the latter case, we applied near-lossless texture compression, using the default texture coding method in V-DMC, on top of the reconstructed geometry for all the examples in Figure 4.12, in order to limit the texture coding artifacts and put

Table 4.1: BD-rates from Figure 4.11 comparing our proposed method with the default HEVC displacements coding in V-DMC.

C1-ai – lossy geometry [all-intra]				
Class	**Sequence (_voxelized)**	**D1 PSNR [194]**	**D2 PSNR [194]**	**IBSM Geom. [56]**
cat1-A	longdress	3.0%	3.0%	2.9%
	soldier	2.9%	3.0%	3.1%
	average	**2.9%**	**3.0%**	**3.0%**
cat1-B	basketball_player	3.1%	3.1%	3.2%
	dancer	2.7%	2.7%	2.7%
	average	**2.9%**	**2.9%**	**2.9%**
cat1-C	football	0.1%	0.1%	-0.1%
	levi	2.9%	2.6%	0.8%
	mitch	1.4%	1.4%	1.4%
	thomas	1.2%	1.2%	0.3%
	average	**1.4%**	**1.3%**	**0.6%**
Overall average		**2.2%**	**2.1%**	**1.8%**

the focus on the *geometry* differences from each method. For the rendered images with texture, we also turned lighting on, to make the geometry differences more discernible.

In Figure 4.12, we can see the visual benefit of the quality scalability provided by our zerotree solution—an increasingly smoother and more accurate geometry reconstruction, achieved *without changing the mesh spatial resolution*. Importantly, we can also observe that even before reaching a lossless displacement reconstruction (lossless with respect to the maximum quality possible for the target CTC rate point), one or more of the lossy reconstructions that we can achieve at a (much) smaller geometry bitrate can still be visually acceptable. For V-DMC applications focused on human visualization, lossless displacement reconstruction at each rate point will likely be unnecessary, as exact geometric precision is often imperceptible, especially at higher CTC rates. Unlike our method, the HEVC and AC displacements coding methods in V-DMC lack quality control, encoding meshes only at a single bitrate and quality level per rate point.

Fairly comparing the complexity of our proposed method with anchor-HEVC is difficult, due to the anchor's lack of scalable coding and due to the fact that HEVC is already a highly developed codec, whereas our current implementation is only a first prototype. However, to achieve the same D1-PSNR (Figure 4.11), our current implementation is on average 1.1x slower for encoding and 12.6x slower for decoding at r01. At r05, the encoding time remains similar, at 1.2x slower, while decoding time increases, to 35.9x slower. Significant improvements are still possible in our code. The main bottleneck is in generating and processing the edge-based hierarchy data, which is currently done sequentially, but these operations are highly parallelizable. Additionally, suboptimal

Figure 4.12: Examples of progressive geometry reconstructions from our zerotree method, using (left to right in each row): 1, 2, 3, 4, 5 zerotree passes corresponding to "zt-r01" from Figure 4.10, compared to the non-progressive reconstruction using HEVC-encoded displacements at the same CTC rate point (r01), and the ground truth (original input mesh) shown at the far right. The red rectangles outline regions where geometric changes are particularly noticeable. Corresponding reconstructions with near-lossless texture compression applied on top are shown underneath.

data structures are currently used, with occasional data duplication or recomputation for easier management across zerotree passes. These inefficiencies can be optimized to substantially reduce memory usage and processing times.

We believe that the proposed zerotree coding method would be even more effective when there is a larger number of subdivision levels in the hierarchy and/or when the base mesh is simpler. Then there would be fewer zerotrees in total and they would be longer, leading to longer hierarchical sequences of wavelet coefficients that are more easily compressible.

4.4 CONCLUSIONS

In this chapter, we presented the first scalable, embedded solution for encoding the *subdivision wavelet coefficients* in the current state-of-the-art codec for dynamic time-varying mesh compression: the MPEG V-DMC. The proposed solution applies zerotree coding on top of an edge-based tree hierarchy that is constructed in 3D space across different subdivision levels, in each frame of a dynamic mesh sequence. The "base mesh displacements" are encoded separately. Given a base mesh and a target subdivision surface, our codec produces a smooth progression of lossy-to-lossless geometry reconstructions, with very little input from the user. This adds the desirable functionality of *quality scalability* of the mesh geometry, which is currently missing from the V-DMC framework. The embeddedness of our bitstream enables the encoder and decoder to tailor the bitrate and quality of the displacements bitstream according to their needs. We believe that such capabilities would be important to ensure the applicability of the V-DMC standard in a wide range of use cases and systems. Our proposed solution and the associated self-contained embedded bitstream have a very simple high-level parameterization that makes the solution easily integrable into the V-DMC software (or any other system that contains subdivision wavelet coefficients). Also, differently to the known existing methods that use zerotree-like coders for subdivision wavelet coefficients on *static* meshes, our proposed method utilizes zerotree coding directly in 3D space, without any prior set partitioning of the wavelet coefficients (most of the other existing methods use image-based SPIHT coding).

5 OPTIMIZING BIT-ALLOCATION OF MULTIPLE 3D ASSETS FOR IMMERSIVE-VIEWPOINT APPLICATIONS

In scenarios such as 3D conferences, numerous assets must be seamlessly merged and rendered together in a single coherent 3D scene, such as illustrated in Figure 5.1. These assets may be captured using different devices, from various locations, and often require integration into the same interactive environment in real-time. It is, therefore, essential that efficient compression and transmission methods are employed to facilitate the implementation of such applications.

Figure 5.1: Example of an immersive teleconference with multiple 3D remote assets.

Several approaches can be used to help the delivery of volumetric video [197]–[199]. User adaptation strategies involve taking into account what the user is actually seeing to adapt the delivery of content [200]–[212]. A uniform treatment of data from different sources and at different positions, in the 3D scene, can lead to inefficient bandwidth use and to lower perceived quality, as more relevant parts of the scene, which are critical to quality of experience (QoE), are not prioritized. Additionally, significant portions of the data may remain unseen at any given moment—such as parts of an object that are occluded by other 3D objects, or outside the user's field of view. Leveraging the user's position and field of view while visualizing volumetric content can substantially reduce network usage without noticeably affecting visual quality.

Different techniques have been developed to account for the user's viewpoint, e.g., tiling, LoD adjustments, view-dependent streaming, and sub-sampling [199], [202], [204], [205], [209], [213],

[214]. However, many of these approaches are overly complex and fail to fully address the combined challenges of dynamic user behavior, occlusion, and the multitude of factors influencing QoE.

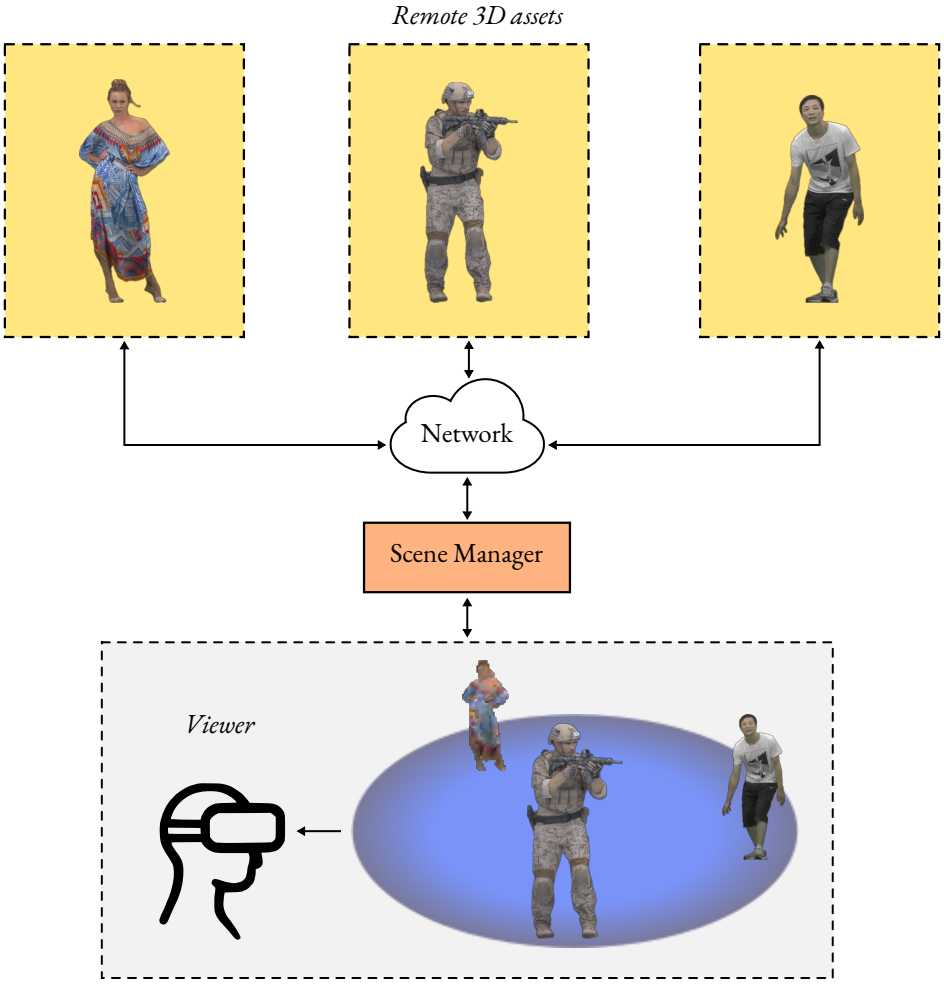


Figure 5.2: Simplified scheme of the framework utilized for the simulations. The proposed solution aims to optimize the bitrate allocation between the remote 3D assets and the scene manager depending on the user's viewpoint.

In this chapter, we propose a viewpoint-aware strategy for optimizing bitrate allocation across multiple 3D assets in the uplink of a simulated 3D meeting environment. This setup is implemented using a scene manager framework, illustrated in Figure 5.2. Although our work does not model the downlink or explore deployment constraints, the scene manager could conceptually operate either remotely (e.g., in a split-rendering paradigm [213], [215]) or locally alongside the viewer. Each 3D asset independently transmits its compressed bitstream (as a mesh or point cloud) to the scene manager, which then decompresses the data and assembles a coherent 3D scene. A 2D projection is rendered from this scene according to the viewpoint requested by the viewer. Finally, the rendered 2D frame is delivered to the user in an interactive streaming fashion. Given the complete scene configuration, we aim to answer the question of what bitrates should the scene manager negotiate with each asset in order to minimize the observed image distortion. Other aspects, such as transmission protocols, compression pipelines, or latency handling, fall outside the scope of this work.

5.1 RELATED WORK

Rate-distortion optimization (RDO) has long been a cornerstone of video compression research, providing a mathematical framework to balance compression efficiency and quality, as seen in Section 2.3. Building on this, a class of methods has emerged that optimize interactive videos by accounting for the user's viewpoint and focus, as demonstrated in FTV systems [216], [217]. More recent efforts have been put into user-adaptive RDO for volumetric video. Adaptive rate allocation and MPEG-DASH-compliant frameworks for point cloud streaming were proposed in [201], utilizing spatial sub-sampling and rate adaptation techniques to prioritize objects based on their proximity to the user's viewport. In [204], a dynamic adaptive streaming system was proposed to improve the delivery latency of static objects for AR, better LoDs were streamed only when necessary either by entering into or getting closer to the user's viewport. In [202], [203] a rate-utility optimization framework to adapt LoDs for AR was proposed, using 3D tiles whose LoDs were dynamically adjusted based on their relation to the user's view frustum and distance to the camera. Optimizing network conditions was also considered, and a window-based client buffer manager was introduced to improve responsiveness over traditional buffer queues. These efforts were extended in [205], where different heuristics were employed to perform 3D tile-based RDO of compressed point clouds. They proposed to adapt the tiles based on a ranking depending on distance, visible area of the bounding box, potential visible area and the number of bits of each representation. Their RDO problem was solved using three different approaches: greedy, uniform and hybrid. More recently tile-based adaptive point cloud streaming was further extended in [212], where they proposed a QoE-driven framework, integrating a novel QoE metric with submodular optimization to maximize user satisfaction under bandwidth constraints. Point Cloud Multipoint Control Unit (PC-MCU) [206] was introduced to optimize multi-user holoconferencing, addressing client-side computational and bandwidth challenges.

5.2 RATE-DISTORTION MODELING AND OPTIMIZATION

5.2.1 COMPRESSION CONTROL STRATEGY

We define a *scene* as a collection of 3D assets (or objects), placed in a common space where they can interact with each other. In practice, controlling bitrate in 3D codecs (e.g., G-PCC and V-DMC) requires tuning multiple compression parameters, many of which are interdependent and do not always yield valid or meaningful results across all combinations. These codecs typically rely on a wide range of encoding settings to be specified, introducing significant complexity to the RDO problem. In order to simplify this, we precomputed RD points using a range of encoding configurations.[1] In this way, instead of tuning multiple low-level parameters, our application simply needs to select the most suitable RD point from a set of predefined representations. These represen-

[1] Configuration files for the utilized codecs can be found at https://datacloud.hhi.fraunhofer.de/s/ry6ZtStJ5Kao2Di.

tations differ primarily in their geometry and texture quantization parameters (QPs), along with other codec-specific settings that also influence bitrate and distortion.

Figure 5.3 illustrates an example of the calculation of such representations for the *longdress* point cloud using G-PCC v23.0 with the octree-RAHT configuration. In this case, 16 geometry QP values were combined with 16 texture QP values to generate a range of RD points. The subset of points forming the lower convex hull (in black) represents the optimum trade-offs between bitrate and distortion. Since 3D assets can significantly vary in the number of points (or faces) they contain, directly comparing their total size in bits is not straightforward. Larger assets will naturally require more bits due to their higher point count, which does not necessarily correlate with better quality. In order to address this, we use bits per input voxel (bpiv) and bits per input face (bpif) as more meaningful metrics for point clouds and meshes, respectively. These measures provide a normalized way to assess how efficiently bits are used to enhance quality, regardless of the asset's size. The adopted distortion metrics, previously discussed in Section 2.4.2, were the PMSE for point clouds, and the IBSM for meshes.

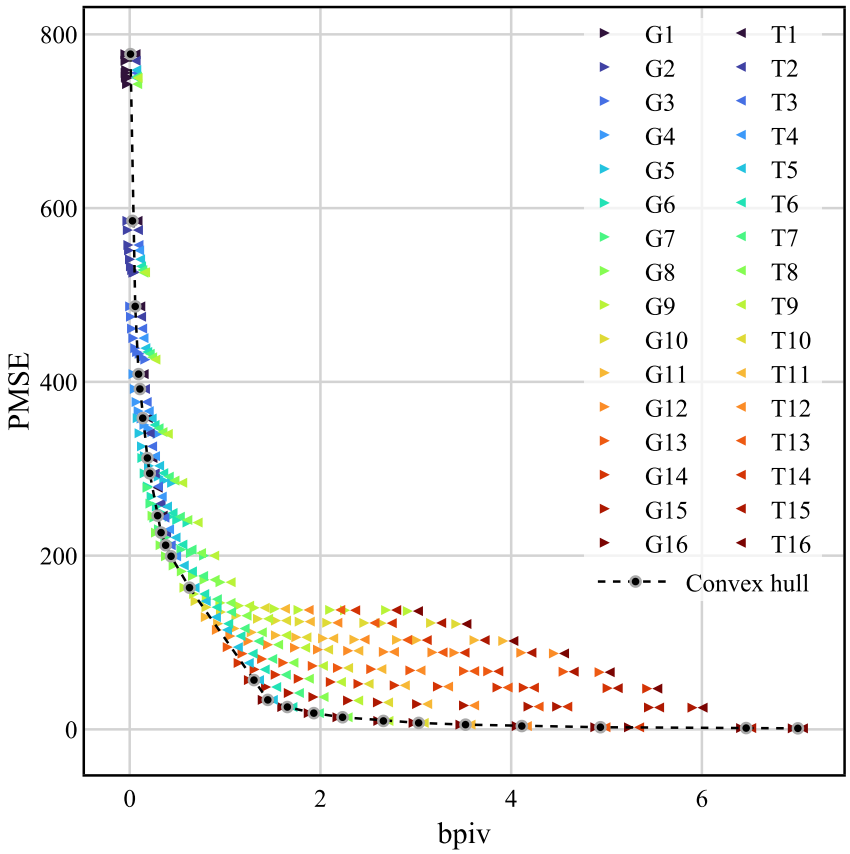


Figure 5.3: PMSE *vs.* bpiv (bits per input voxel) for *longdress* using G-PCC. We collected rate-distortion data over 16 QPs for geometry (G) and texture (T), then we calculated the convex hull of those points.

Therefore, we map all the K encoding parameters for an asset in a tensor, such that the m-th representation is achieved by having

$$\mathbf{Q}(m) = [\mathrm{QP}_{\mathrm{geo}}, \mathrm{QP}_{\mathrm{tex}}, par_1, par_2, ..., par_{K-2}]. \tag{5.1}$$

Then, the manager needs to choose a different RD point m from the set of available representations.

5.2.2 Distortion Modeling

For the targeted applications, 3D content is ultimately projected onto 2D images, as conventional displays remain the primary medium for end-user visualization. Since we only need to measure distortion from specific viewpoints, traditional 3D distortion metrics are not ideal. When assets have different spatial resolutions, point-wise distances can be misleading, as variations in sampling density introduce artificial discrepancies. As a result, metrics such as point-to-point (see Section 2.4.1) serve as a poor proxy for viewpoint-based distortion.

Considering a scene with N assets, each asset i is rendered using the m_i-th representation, which is one of the M different representations. Besides the representation choice, the individual distortion D of the i-th asset, as seen by the viewer, also depends on its position and the camera parameters used to capture the scene at a given viewpoint v, such that

$$D_i = D_i(m_i, \mathbf{X}_i(v), \mathbf{P}(v)), \tag{5.2}$$

where,

- m_i indicates the m-th representation is chosen for asset i;
- $\mathbf{X}_i(v)$ represents the 3D coordinates of the object i in world space at viewpoint v;
- $\mathbf{P}(v)$ is the camera projection matrix, which includes both the intrinsic and the extrinsic parameters, and the camera translation vector.

We want to model the total distortion $\mathcal{D}$ from a viewpoint v, considering a set of chosen compressed versions

$$\mathbb{M} = \{m_1, m_2, \ldots, m_N\} \tag{5.3}$$

for each of the N assets in the scene,

$$\mathcal{D}(\mathbb{M}, v) = \sum_{i=1}^{N} D(m_i, v). \tag{5.4}$$

We assert that (5.2) can be simplified by considering two main components: *compression distortion*, which solely depends on the chosen compressed representation m_i; and *view-dependent distortion*, which depends on the camera configuration and viewpoint v, shaping how the chosen representation is perceived when rendered. In particular, since we are combining multiple assets in a single scene and analyzing their effect on the projected 2D image, MSE offers a convenient and widely used measure that supports straightforward comparison across projections. To make this metric independent of the camera, we used PMSE for point clouds and IBSM for meshes (see Section 2.4.2). IBSM is used here for convenience, since it is readily available in the V-DMC TM

software. We argue that we can use projected metrics to measure the compression distortion effects from the 3D codecs in (5.2). We refer to this measured distortion as $\tilde{D}(m_i)$.

In order to take into account the view-dependent distortion effects from $\mathbf{X}_i(v)$ and $\mathbf{P}(v)$ in (5.2), we introduce an importance measure $\alpha_i(v)$ to ponder $\tilde{D}(m_i)$ according to the specific position and camera configuration of each asset in the scene. By combining these two factors, we can have a proxy for the total distortion (5.4) as:

$$\mathcal{D}(\mathbb{M}, v) \propto \tilde{\mathcal{D}}(\mathbb{M}, v) = \sum_{i=1}^{N} \alpha_i(v)\tilde{D}(m_i). \tag{5.5}$$

Note that this importance formulation targets visual-quality applications; geometry-critical tasks may require a different measure.

5.2.3 Importance Measure

Deciding what is important in a scene is highly subjective and depends on the application. For example, importance might be more strongly weighted towards a person speaking in a video conference, an object that is close to the camera, or areas that occupy a significant portion of the viewer's field of view. The distortion model presented in this work is agnostic to how this importance measure is calculated, as long as $0 \leq \alpha_i(v) \leq 1$ and $\sum_i \alpha_i(v) = 1$, for each v.

Previous works [201], [202], [205] have considered factors such as the Euclidean distance of an asset from the camera and its projected area as relevant indicators for bitrate allocation. We also advocate to use an importance measure based on the occupied area (in pixels) of each asset in the projection of the viewpoint; we believe that this is a good generalization of the importance problem for most applications. We reason that this measure takes into account the camera parameters, the distance between assets and the camera, occlusions, and voxel distinguishability[2] as voxels get smaller (more than one voxel will be projected to a single pixel, making their area smaller). To allow fairness in the measure for assets with very distinct sizes (e.g., an adult and a child), we normalize the area occupied by each asset by the length of the diagonal of its bounding box, such that

$$\alpha_i(v) = \frac{\frac{\texttt{pixel\_count}_i(v)}{\texttt{diagonal}_i}}{\sum_{i=1}^{N}\left(\frac{\texttt{pixel\_count}_i(v)}{\texttt{diagonal}_i}\right)}. \tag{5.6}$$

5.2.4 Bit-Allocation Optimization

We want to select the set $\mathbb{M}$ of optimal representations for the N assets which minimize the scene's total distortion at each viewpoint v, as in (5.5), subject to

$$\mathcal{R}(\mathbb{M}) = \sum_{i=1}^{N} R(m_i) \leq \mathcal{R}_b, \tag{5.7}$$

[2] Although display resolution, pixel density and the viewing distance also influence voxel distinguishability, we simplify the discussion by assuming constant viewing conditions and displays characteristics.

where $\mathcal{R}_b$ is a given budget bitrate. Since the overall bitrate and distortion are additive across assets, and the RD points are usually bound by a well-behaved curve, we tackle the above problem by defining a Lagrangian cost function,

$$\begin{aligned} \mathcal{J}(\mathbb{M}, v) &= \mathcal{R}(\mathbb{M}) + \lambda \tilde{\mathcal{D}}(\mathbb{M}, v) \\ &= \sum_{i=1}^{N} R(m_i) + \lambda \alpha_i(v) \tilde{D}(m_i), \end{aligned} \tag{5.8}$$

where, λ is a Lagrange multiplier that balances the trade-off between rate and distortion, in our search for the minimum cost, as discussed in Section 2.3. Moreover,

$$\begin{aligned} &\min_{\mathbb{M}} \left[\mathcal{R}(\mathbb{M}) + \lambda \tilde{\mathcal{D}}(\mathbb{M}, v) \right] = \\ &\min_{\{m_1, \ldots, m_N\}} \left[\sum_{i=1}^{N} R(m_i) + \lambda \alpha_i(v) \tilde{D}(m_i) \right] = \\ &\sum_{i=1}^{N} \min_{\{m_1, \ldots, m_N\}} \left[R(m_i) + \lambda \alpha_i(v) \tilde{D}(m_i) \right]. \end{aligned} \tag{5.9}$$

Hence, we reduce the problem to minimize the Lagrangian function for each individual asset. By making $\lambda_i(v) = \lambda \alpha_i(v)$ as the individual Lagrangian multiplier for asset i at viewpoint v, its optimal representation $m_i^*\big|_{\lambda_i(v)}$ can be found by solving

$$\begin{aligned} m_i^*\big|_{\lambda_i(v)} &= \arg\min_{m_i} [J(m_i)] \\ &= \arg\min_{m_i} [R(m_i) + \lambda_i(v) \tilde{D}(m_i)]. \end{aligned} \tag{5.10}$$

Note that, since $0 \leq \alpha_i(v) \leq 1$, we have that $0 \leq \lambda_i(v) \leq \lambda$.

Solving (5.10) becomes straightforward once the values of $R(m_i)$ and $\tilde{D}(m_i)$ are precomputed (or possibly estimated). For each asset and for each viewpoint, we calculate $\alpha_i(v)$, and then select the representation that has the lowest associated cost.

Note, also, that the case of uniform allocation implies

$$\alpha_i(v) = \frac{1}{N}, \ \forall \, i, v, \tag{5.11}$$

i.e., uniform $\lambda_i(v)$, as per design.

For the proposed formulation, λ_i represents the negative reciprocal of the slope of the convex hull in the RD plane. As we increase the value of λ_i from 0 to ∞, we cover all the points in the convex hull, from the low-rate-high-distortion to high-rate-low-distortion, respectively (see Figure 2.9(a)). Given the finite number of compressed versions that can be produced, the parameter λ_i is, in practice, discretized in accordance with the achievable RD points associated with the respective curve for each asset.

5.3 EXPERIMENTS

5.3.1 SETUP

Figure 5.4 illustrates the details of the scene manager we envisioned for the experiments. For each frame, the manager decodes each incoming 3D asset and uses the provided metadata to assembly a 3D scene. Based on the user-specified viewpoint, it renders a 2D projection of this scene and calculates the importance value $\alpha_i(v)$ for each asset for the current viewpoint.

Given a Lagrange multiplier λ, the system then solves the optimization problem from Section 5.2.4 to determine the optimal QPs for each asset. These encoding settings are sent back to the remote sources, after which the compressed assets are transmitted to the scene manager accordingly. Here, for simplicity, we considered a clairvoyant prediction of the user's viewpoint, such that every frame is optimized for the *current* position, without delay.

The manager then renders a 2D frame and sends it to the user. If the user requests a new viewpoint, the updated parameters are fed back into the system, triggering a new round of importance evaluation, QP selection, and rendering. This setup enables the evaluation of different importance weighting strategies under dynamically changing viewpoints.

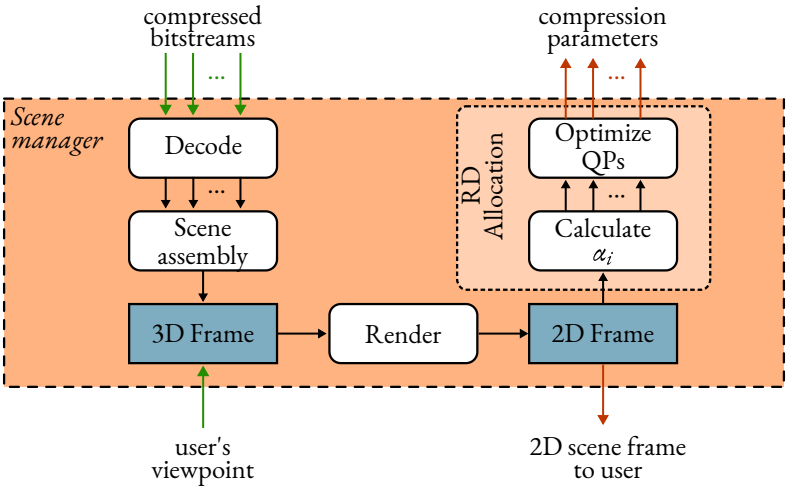


Figure 5.4: Framework of the scene manager used for the simulations.

Each asset includes metadata to ensure correct placement in the scene, such as geometry resolution (bit depth), frame-to-world scale, bounding box dimensions, and indicators for vertical and forward axes. The geometry resolution and frame-to-world scale enable the coherent merging of assets, providing visual consistency and realism to the scene. Assuming a prior negotiation to determine broadcast resolution, we normalized assets to a common scale and resolution, opting for the lowest resolution after scaling to reduce complexity. The bounding box dimensions are used in positioning and calculating centroids for distance measures, while the vertical and forward axes ensure proper orientation of the assets.

For the scene assembly, the dimensions of the virtual stage must also be defined, as well as the

type of 3D assets (meshes or point clouds). After assembling each 3D frame, user viewpoints are used to calculate the importance measures of each asset. Figure 5.5 illustrates the areas used for the importance computation.

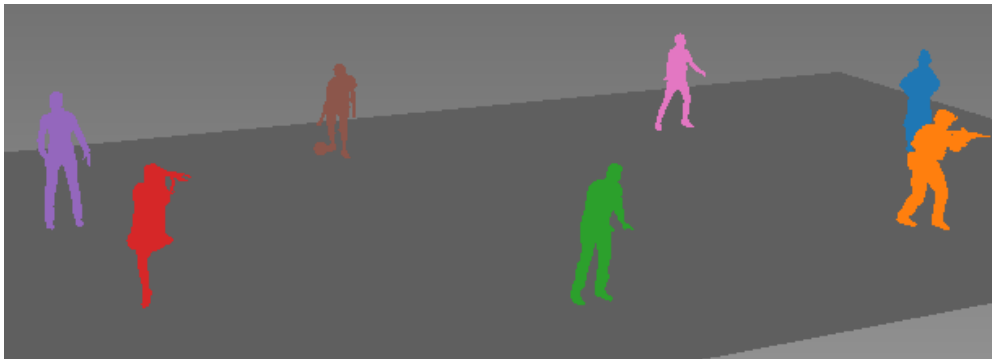


Figure 5.5: Illustration of the areas used for the calculation of the importance measure. Since the assets are segmented, only the visible pixels from each asset are considered for each viewpoint. The dark gray rectangle area represent the scene stage size, as defined by the scene manager.

Point cloud and meshes tested were human figures from MPEG's G-PCC and V-DMC respective CTCs [218], [219]. Table 5.1 provides a summary of the tested assets.

5.3.2 Tests

To better understand the proposed approach, three tests were performed with increasing complexity, as illustrated in Figure 5.6:

1. single static asset repeated at different positions (point clouds),
2. multiple static assets (point clouds),
3. multiple dynamic assets (meshes).

Additionally, seven strategies for rate allocation were tested:

i) equal rate for all assets (no optimization is performed),

ii) proportional rate according to the visible area of each asset (no optimization is performed),

iii) uniform distribution of importance,

iv) aware uniform distribution (only visible assets are considered),

v) distance-based importance,

vi) aware distance-based importance,

vii) area-based importance, from (5.6).

The first test serves as a control. Since the same asset is repeated at different positions in the scene and we can clearly observe how the rate allocation is done depending on the position of the

Table 5.1: Summary of tested assets.

Point clouds	geo. res.	# voxels	scale [cm]
(a) Group: *8i_vox10* [175]†			
longdress_vox10_1300	10-bit	857,966	0.180
loot_vox10_1200	10-bit	805,285	0.182
redandblack_vox10_1550	10-bit	757,691	0.182
soldier_vox10_0690	10-bit	1,089,091	0.182
(b) Group: *owlii* [177]‡			
basketball_player_vox11_00000200	11-bit	2,925,514	–
dancer_vox11_00000001	11-bit	2,592,758	–
(c) *queen_frame_0200*‡	10-bit	1,000,993	–
Meshes	**geo. res.**	**# vertices**	**# faces**
(a) Group: *8i_vox10* [175]†			
longdress	10-bit	22k	40k
soldier	10-bit	22k	40k
(b) Group: *owlii* [177]‡			
basketball	12-bit	20k	40k
dancer	12-bit	20k	40k
(c) Group: *Volucap and XDProd*‡			
mitch	12-bit	16k	30k
thomas	12-bit	16k	30k
football	12-bit	25k	40k
levi	12-bit	20k	40k

† https://jpeg.org/plenodb/

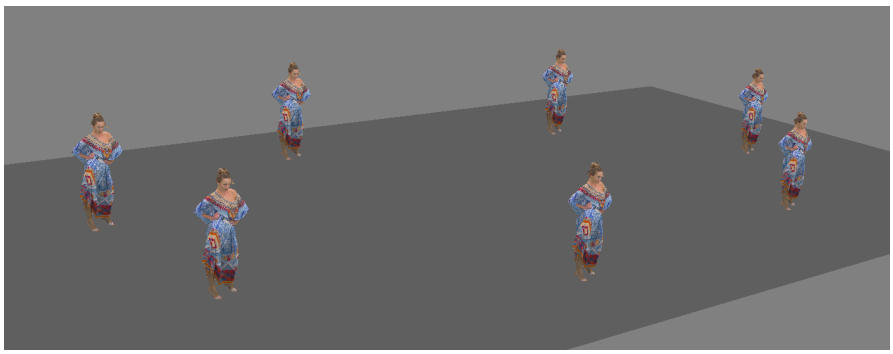
‡ https://mpegfs.int-evry.fr/mpegcontent/

asset in respect to the user's point of view, as illustrated in Figure 5.6.

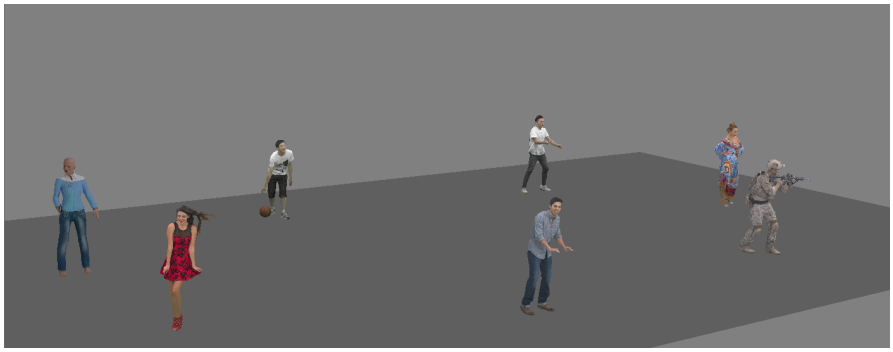
For the first two tests, point clouds were used and the compression was performed using G-PCC v23.0. In G-PCC, QP_{geo} actually scales down the octree, which is then rescaled back to its original size after the removal of duplicate points (see Section 2.5.1). This operation can make the point cloud very sparse, visually this is usually solved by rendering the point cloud using larger voxels. However, since we may have point clouds with different resolutions in a scene, we interpolated the missing points using the NNI (see Section 3.2.2) before rendering. This interpolation gives a blocky aspect to the asset, visually it is the same as enlarging the voxels but independently for each asset.

As the camera zooms and pans through the scene, the number of visible assets gradually decreases.[3] The importance measure of each asset in each frame is depicted in Figure 5.7. The im-

[3]Videos for the camera paths and results for all tests can be found at https://datacloud.hhi.fraunhofer.de/s/ry6ZtStJ5Kao2Di.



Test 1



Test 2

Test 3

Figure 5.6: Visualizations of the three tests scenarios. Test 1: repetition of a single static asset. Test 2: multiple static assets. Test 3: multiple dynamic assets.

portance measure from the "Uniform Aware" strategy actually reflects the number of visible assets in each frame. Frames where there is a change in the number of visible assets are marked with dotted lines. Notably, the *aware* strategies consider an asset visible as long as at least one pixel of its projection is detected.

The camera path consisted of 100 frames, and tests were conducted using 6 different target rates. Figure 5.8 shows BD-rate and BD-PSNR for each frame using the uniform distribution as the reference. In those plots, the frames where the number of visible assets changes are marked by a dotted line. Looking also at Figure 5.7, it is possible to compare how the different strategies cope with assets leaving or entering the visible area.

Additionally, Figure 5.9 displays the average RD points across all frames that are rate targets. The "Equal Rate" yields results similar to the uniform distribution, but slightly worse, since it may

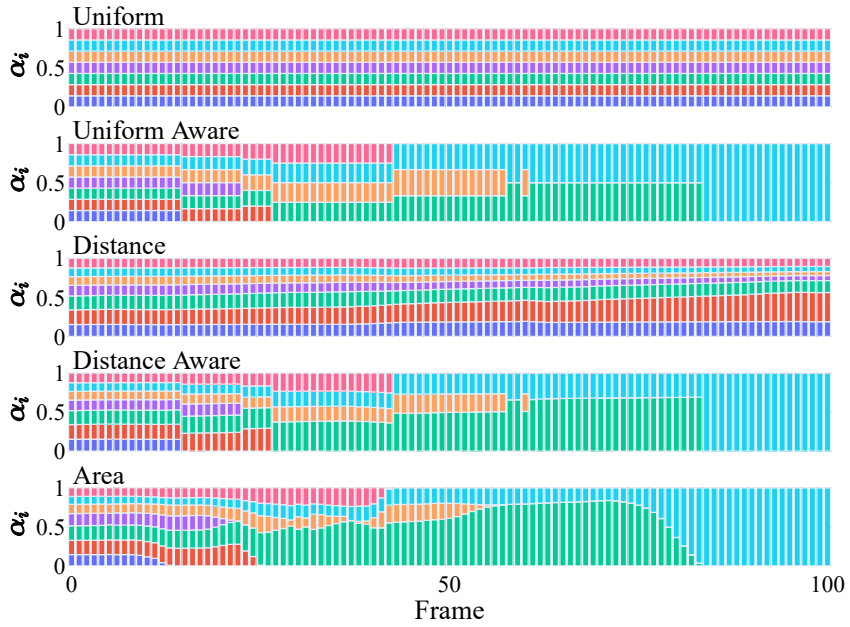


Figure 5.7: Importance measure of each asset per frame in Test 1. Dotted lines mark the frames at which assets enter or exit the camera's view frustum.

select non-optimized RD points. This issue is even more pronounced in the "Proportional Rate" approach, where representations were selected solely based on rate, disregarding distortion. As a result, when asset proportions were similar, it performed poorly (first frames), only surpassing the uniform anchor once the number of visible assets was nearly halved—suggesting that its gains stem mostly from asset awareness. The oscillations in this approach are due to the fact that it was set to select the minimum rate to satisfy the proportionality of visible area, therefore small changes in the areas caused the selection of new (non-optimal) representations. The benefits of visibility awareness are more evident in the "Uniform Aware" and "Distance Aware" approaches, which show significant improvements over their non-aware counterparts. It is interesting to see that while distance can serve as a useful importance measure, it fails to outperform the uniform distribution unless combined with camera orientation. This is further illustrated in Figure 5.7, where, in the final frames, a shorter distance to the camera did not align with the actual projected asset size. Finally, the "Area" approach achieved the best results, improving upon "Distance Aware" by incorporating occlusions and projected sizes—key factors for the distortion determination.

When comparing Figure 5.8 and Figure 5.9, it is clear that the results are highly dependent on the camera path. Although the chosen camera path is somewhat arbitrary, it can be interpreted as a combination of simpler path segments, each representing different typical viewpoint configurations. As a result, it serves to illustrate a range of user's viewpoints within a single experiment. However, average results derived from this trajectory should be interpreted with caution, as the prominence of specific segments could significantly influence the overall results. For instance, if all assets were visible throughout all frames, the difference between the methods would have been significantly smaller. This suggests that no single method is universally optimal in all scenarios. Still, the proposed framework allows for performance assessment under different conditions and

can adapt to varying importance methods as needed.

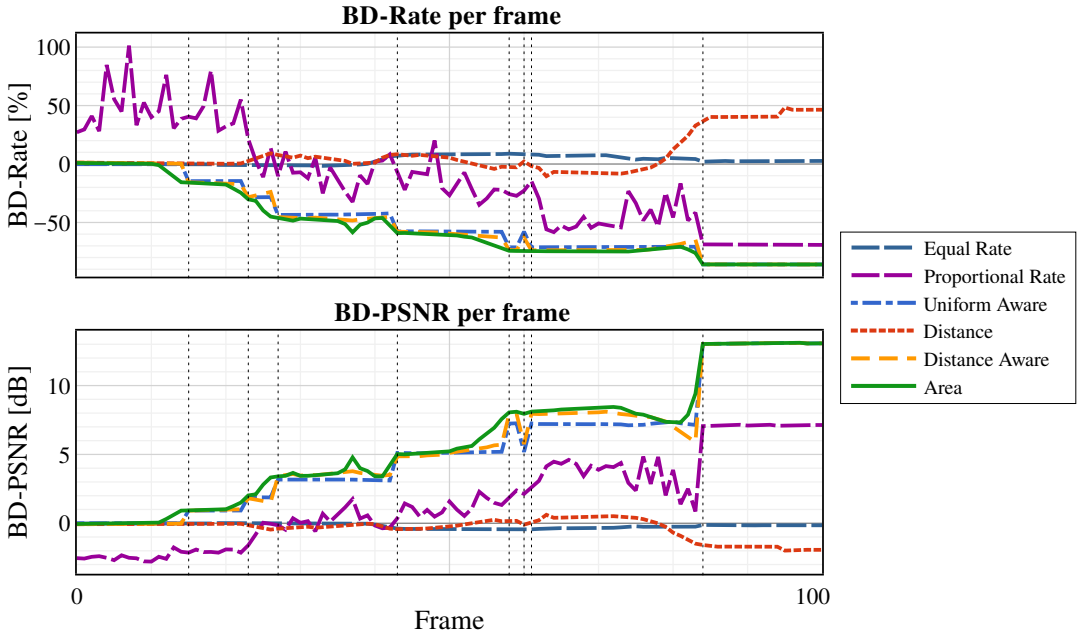


Figure 5.8: BD-rate per frame Test 1, using the performance of the uniform importance strategy as reference. Dotted lines mark the frames at which assets enter or exit the camera's view frustum.

Table 5.2: Average BD-metrics for Test 1.

	BD-Rate [%]	BD-PSNR [dB]
Equal Rate	3.99	-0.23
Proportional Rate	-23.69	1.16
Uniform Aware	-55.34	4.65
Distance	5.02	-0.26
Distance Aware	-57.62	4.97
Area	-59.27	5.17

For the second test, all the point clouds from Table 5.1 were placed into the scene. The same camera path and the same position of the assets from Test 1 were used. Meaning that the importance measures were pretty similar to the ones in Figure 5.6, with only small differences due to asset's size and pose.

The results in Figure 5.10 and Figure 5.11 show a behavior similar to that observed in Test 1, with one notable exception as the "Equal Rate" approach performed significantly worse in this case. This decline in performance is due to the differences in intrinsic distortion curves across assets. Allocating the same bitrate to all assets, without accounting for distortion, leads to severe quality degradation, highlighting the importance of distortion-aware allocation strategies.

For the third test, the dynamic meshes from Table 5.1 were employed and a new camera path was defined. The codec of choice was V-DMC v9.0. This time, the camera path is more complex,

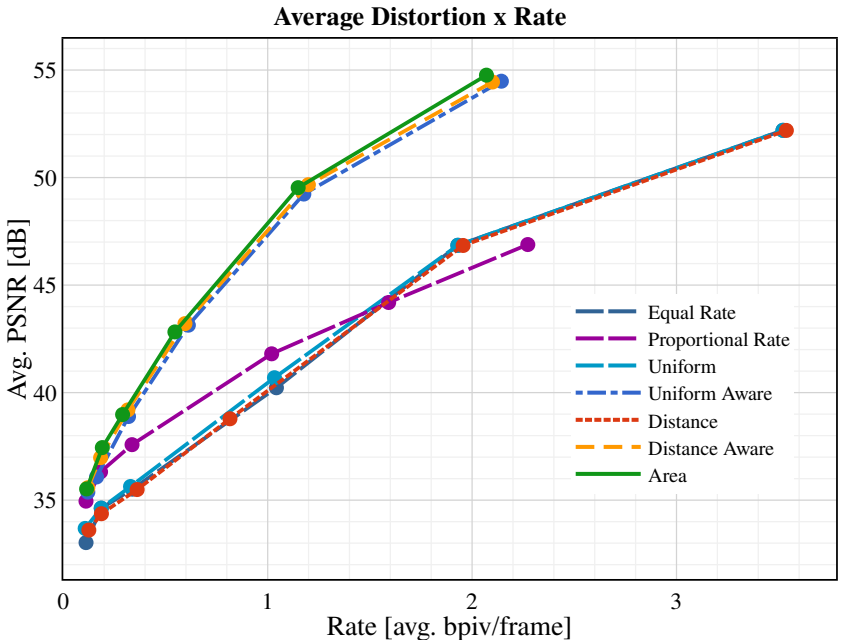


Figure 5.9: Average Distortion vs Rate for Test 1.

Table 5.3: Average BD-metrics for Test 2.

	BD-Rate [%]	BD-PSNR [dB]
Equal Rate	117.35	-2.82
Proportional Rate	-13.61	1.39
Uniform Aware	-57.04	5.41
Distance	5.92	-0.34
Distance Aware	-59.77	5.80
Area	-62.87	6.27

and tries to mimic what a real user would do in such immersive environment, first checking the whole scene then checking a specific asset which they are more interested in.

Figure 5.12, Figure 5.13 and Table 5.4 show that the results from this test follow the same trend as the previous ones, with "Equal Rate" and "Distance" strategies performing poorly. The "Distance" strategy, however, showed an improvement and outperformed the anchor. This is mainly due to the camera path, which kept most of the assets visible for a larger number of frames. This fact also explains the reduced gains seen in the "Area" method, when comparing to the previous tests.

As discussed in Section 2.3.1, BD calculations rely on polynomial fitting, which require well-behaved monotonically-increasing RD points to ensure a good fit. However, due to the nature of the problem we are dealing with, the RD points are not always well-behaved. For instance, some strategies can allocate bits to assets that are not visible, so that an increase in rate would have no improvement in quality. To address this issue, we used piecewise integration to have a more ro-

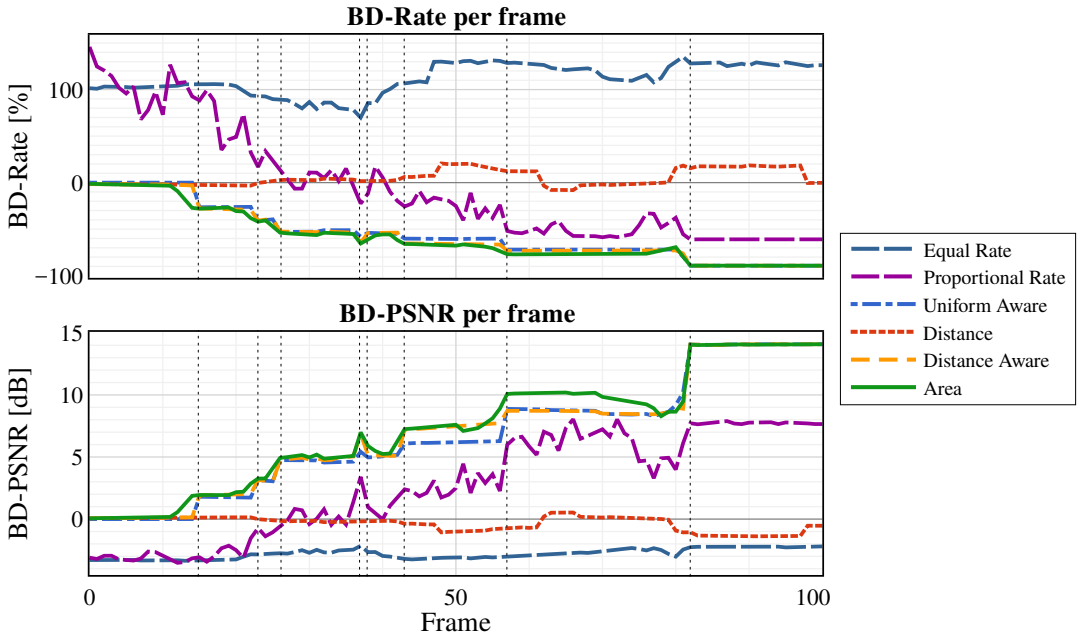


Figure 5.10: BD-rate per frame Test 2, using the performance of the uniform importance strategy as reference. Dotted lines mark the frames at which assets enter or exit the camera's view frustum.

bust calculation of the BD values, and we removed a few RD points, when needed, to ensure the monotonicity of the data. In some cases, however, especially in Test 3, the compared rates were not overlapping—the anchor rate was significantly higher than the "aware" strategies—which prevented the calculation of BD values on that frame. Such behavior can be seen in Figure 5.12, where some gaps in the curves are visible and the huge BD-rate values are observed (meaning a very narrow range of overlapping rates). In such cases, ideally, another anchor should be used. However, for the sake of simplicity and comparability, this was not carried out, as we wanted the same anchor of tests 1 and 2.

Table 5.4: Average BD-metrics for Test 3.

	BD-Rate [%]	BD-PSNR [dB]
Equal Rate	18.60	-0.51
Proportional Rate	-28.67	1.33
Uniform Aware	-41.96	1.67
Distance	-8.96	0.30
Distance Aware	-50.85	2.23
Area	-52.80	2.35

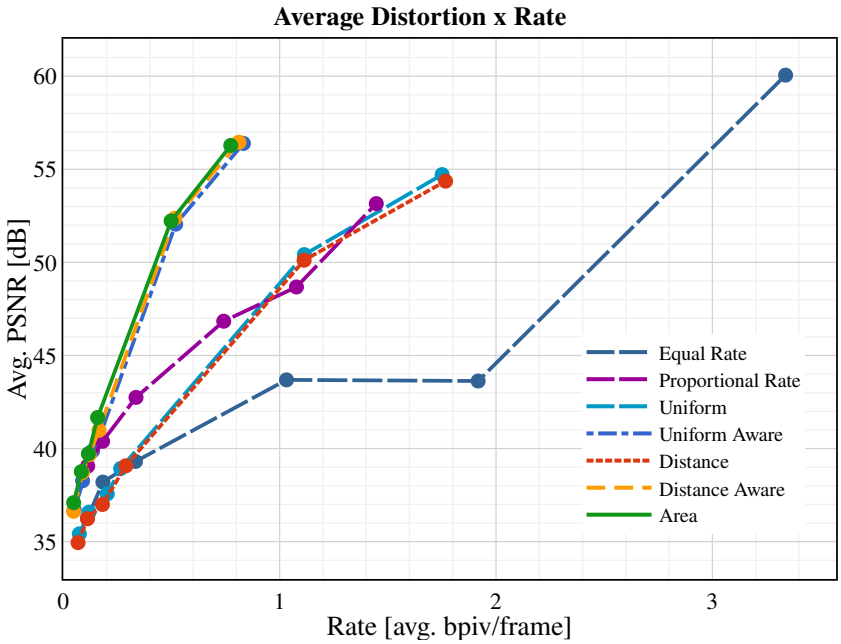


Figure 5.11: Average Distortion vs Rate for Test 2.

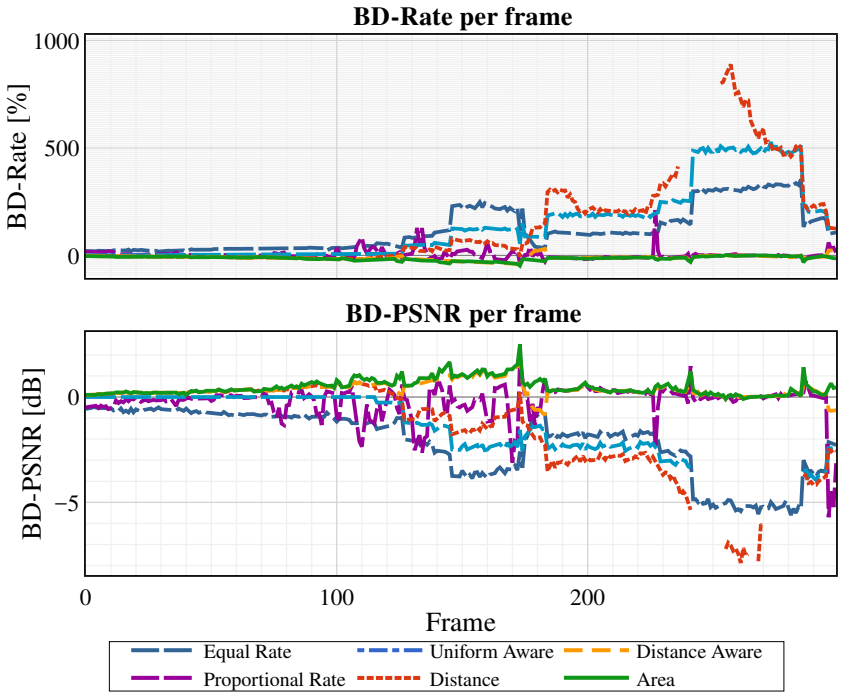


Figure 5.12: BD-rate per frame test 3.

5.4 CONCLUSIONS

In this chapter, we introduced a novel framework for optimizing bit allocation across multiple 3D assets in immersive scenes, based on viewpoint-dependent perceptual importance. Our approach takes into account both *compression* and *view-dependent* distortions, to develop an importance-aware strategy that guides bitrate distribution more efficiently than traditional heuristic

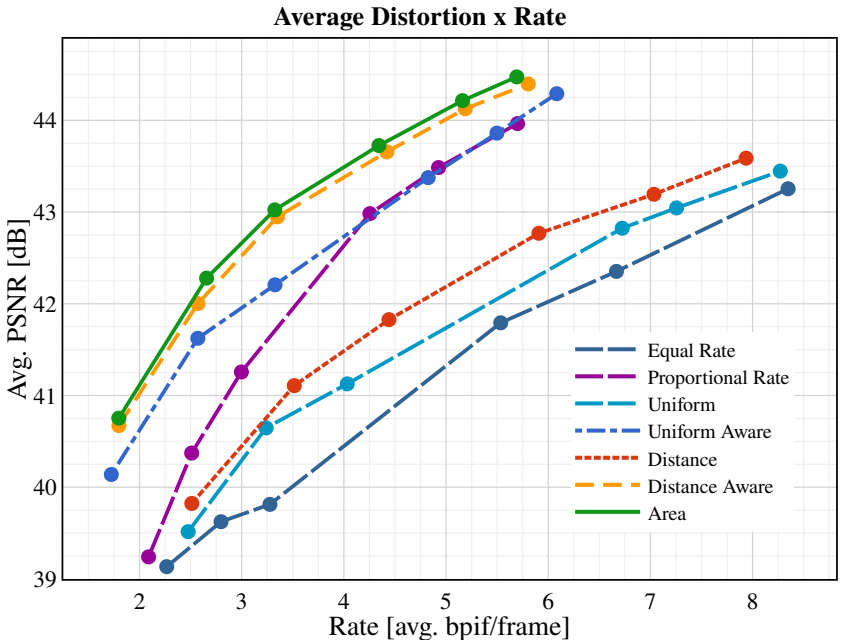


Figure 5.13: Average Distortion vs Rate for Test 3.

approaches.

Through our experiments, we demonstrated how different importance measures can affect the bit allocation and, thus, impact the RD performance. In our tests, the area-based importance outperformed the other strategies achieving average BD-rate savings of at least 52% when compared to the uniform allocation. Since the camera paths used in the experiments are composites of different segments containing a varied number of visible assets, the reported average gains should be interpreted with caution. A different emphasis on specific path segments could lead to substantially different results. Moreover, it was evidenced that the optimal allocation strategy is context-dependent, influenced by factors such as camera movement, scene layout, and system constraints. We also discussed challenges in BD-rate computation for non-monotonic RD curves, highlighting the need for careful benchmarking in adaptive streaming setups.

To the best of our knowledge, this is the first principled method to tackle viewpoint-adaptive bit allocation in multi-asset 3D scenes, bridging perceptual relevance and practical codec integration. These findings confirm that scene-awareness and perceptual importance must be central to bitrate management in volumetric media systems to enhance user QoE.

6 CONCLUSIONS

This thesis explored different aspects of 3D content compression, addressing challenges in point cloud super-resolution, scalable geometry coding for dynamic meshes, and bit allocation strategies for immersive applications. The work began with a background review on Chapter 2, covering key 3D representations, compression standards such as MPEG's PCC and V-DMC, scalable coding techniques, quality assessment methods, and rate-distortion optimization theory.

Chapter 3 focused on a fractional super-resolution method for voxelized point clouds, primarily targeted at fractional scale factors in the range $1 < s \leq 2$. By leveraging self-similarities in point cloud structures, the method reconstructs high-quality versions of downsampled content, improving geometric accuracy beyond traditional nearest-neighbor interpolation and smoothing techniques. Experimental results showed consistent improvements, particularly in point-based distortion metrics. Additionally, the method was integrated as a post-processing step for MPEG's G-PCC codec, demonstrating significant quality gains at higher bitrates. Unlike ML-based techniques, which require pre-trained models tuned for specific bitrates, the proposed method provides a lightweight and adaptable alternative without the need for prior training.

Chapter 4 introduced a scalable, embedded coding approach for geometry of dynamic TVMs within MPEG's V-DMC framework. The proposed method applies zerotree coding directly in the 3D space of the mesh, constructing an edge-based tree hierarchy that spans different subdivision levels. By encoding wavelet coefficients in a progressively refinable manner, the method enables quality scalability, a feature currently missing from V-DMC. This approach allows both the encoder and decoder to dynamically adjust the bitrate and quality based on application needs. Unlike existing image-based wavelet compression schemes, which impose rigid structures on the data, our method preserves the native 3D hierarchy, making it well-suited for practical integration into the V-DMC software.

Chapter 5 focused on rate-distortion optimization for immersive applications, specifically addressing the challenge of optimally distributing bits across multiple 3D assets in a scene. Traditional approaches often allocate bitrate uniformly, without considering viewpoint-dependent relevance or perceptual importance. To address this, we introduced an importance measure that quantifies the perceptual significance of each asset at a given viewpoint. Through experimental analysis, we demonstrated that different importance strategies lead to distinct bit-allocation patterns, directly impacting rate-distortion performance. While an area-based importance measure showed strong results in our tests, the optimal strategy is highly context-dependent, varying with scene configuration, camera motion, and system constraints. The findings suggest that bit allocation should not be solely rate-driven but must incorporate scene-awareness and perceptual relevance to enhance the overall QoE.

6.1 SUMMARY OF CONTRIBUTIONS

The research in this thesis followed three distinct but interrelated directions, largely influenced by the author's work across three different institutions—UnB, InterDigital, and Fraunhofer HHI. At each stage, there was a need to balance fundamental research with practical integration into standardization efforts, leading to contributions that spanned different aspects of 3D content coding.

The first contribution was the development of a fractional super-resolution method for voxelized point clouds, which enhances the quality of downsampled (or compressed) point clouds. Initially explored during the author's Master's research at UnB, this work was later adapted to improve integration with point cloud compression pipelines. The method resulted in a publication in IEEE TIP and another in the SBrT/IEEE ComSoc JCIS, as well as contributions to MPEG 3DGH standardization efforts.

The second contribution focused on geometry coding for dynamic TVMs, particularly within the MPEG V-DMC framework. This research, developed at InterDigital, explores a hierarchical coding approach based on subdivision wavelets and introduces a zerotree-based coding method to efficiently encode wavelet coefficients directly in the native 3D space. The outcomes of this research include a paper recently published in IEEE TIP, a patent application, and multiple contributions to MPEG standardization.

The third contribution centers on rate-distortion optimization for immersive applications, addressing the problem of optimal bit allocation across multiple 3D assets. This work, conducted at Fraunhofer HHI, proposes a model for distributing bitrates efficiently in immersive-viewpoint scenarios, ensuring that resources are allocated to maximize perceived quality depending on the importance of each asset for given viewpoints. The findings are in preparation for publication.

Beyond these technical contributions, the research conducted in this thesis actively contributed to standardization activities, providing technical reports and experimental evaluations to MPEG 3DGH discussions. The diversity of topics covered in this work reflects both the evolving nature of the research field and the practical constraints of working within different research environments. Each contribution was shaped by the specific goals and priorities of the institutions where the work was carried out, demonstrating how academic research can intersect with industry-driven standardization efforts to address real-world challenges in 3D content compression.

6.2 FUTURE WORK

The proposed methods show potential, but further research is needed to better understand their impact and refine their applicability. A subjective evaluation of the SR method for point clouds would provide valuable insights into its visual impact. While objective metrics help assess reconstruction quality, a user study could better validate the effectiveness of the approach and guide refinements based on human perception. Additionally, improving robustness against outlier re-

gions and enhancing attribute reconstruction, such as color and normal recovery, would further increase the applicability of SR in challenging scenarios involving sparse or noisy point clouds.

For dynamic TVMs, multi-resolution and quality-scalable coding is expected to evolve from an optional feature into a fundamental requirement. In volumetric video applications with multiple objects, it will be essential for the codec to encode and decode different parts of the scene at varying spatial and quality resolutions, adapting to network conditions and rendering constraints. The proposed zerotree-based method provides an important step in this direction, but further research is needed to refine its scalability and optimize its integration into practical systems. Beyond geometry compression, scalable texture coding remains an important challenge. Future work could explore techniques such as image-based zerotree coding, leveraging scalable video codecs like Scalable-HEVC, or using mipmap-based approaches, similar to the recent V-DMC adoption in [220]. Investigating hybrid solutions that balance scalability with minimal impact on compression efficiency and computational complexity would be particularly relevant. Extending the proposed method toward inter-frame coding could improve temporal consistency in dynamic mesh sequences. Leveraging motion compensation, hierarchical inter-frame prediction, or temporal wavelet transforms may enhance compression efficiency while maintaining progressive scalability.

For immersive applications with multiple 3D assets, future work could also explore additional strategies to further optimize bit allocation. Methods such as the SR approach proposed in Chapter 3 or the scalable coding solution presented in Chapter 4 could be integrated to improve rate-distortion efficiency while maintaining adaptability across different viewing conditions. Furthermore, real-time adaptation techniques, incorporating dynamic viewpoint prediction and user interaction models, could refine the bit allocation by dynamically adjusting compression parameters based on user behavior and scene complexity. Optimizing allocation for a group of users considering a probabilistic distribution of importance—similar to what was done in [217]—is also an interesting path for future research.

REFERENCES

[1] G. Bruder, F. Steinicke, and A. Nuchter, "Poster: Immersive point cloud virtual environments," in *2014 IEEE Symposium on 3D User Interfaces (3DUI)*. IEEE, Mar. 2014, pp. 161–162.

[2] T. Fukuda, Y. Zhu, and N. Yabuki, "Point Cloud Stream on Spatial Mixed Reality - Toward Telepresence in Architectural Field," in *Proceedings of the 36th International Conference on Education and Research in Computer Aided Architectural Design in Europe (eCAADe) [Volume 2]*, ser. eCAADe 2018, vol. 2. eCAADe, 2018, pp. 727–734.

[3] C. Tommasi, C. Achille, and F. Fassi, "From Point Cloud to BIM: A Modelling Challenge in the Cultural Heritage Field," *The International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, vol. XLI-B5, pp. 429–436, Jun. 2016.

[4] Q. Wang and M.-K. Kim, "Applications of 3D point cloud data in the construction industry: A fifteen-year review from 2004 to 2018," *Advanced Engineering Informatics*, vol. 39, pp. 306–319, Jan. 2019.

[5] X. Yue, B. Wu, S. A. Seshia, K. Keutzer, and A. L. Sangiovanni-Vincentelli, "A LiDAR Point Cloud Generator: from a Virtual World to Autonomous Driving," in *Proceedings of the 2018 ACM on International Conference on Multimedia Retrieval*. ACM, Jun. 2018.

[6] P. Schelkens, T. Ebrahimi, A. Gilles, P. Gioia, K.-J. Oh, F. Pereira, C. Perra, and A. M. G. Pinheiro, "JPEG Pleno: Providing representation interoperability for holographic applications and devices," *ETRI Journal*, vol. 41, no. 1, pp. 93–108, Feb. 2019.

[7] M. Domanski, O. Stankiewicz, K. Wegner, and T. Grajek, "Immersive visual media – MPEG-I: 360 video, virtual navigation and beyond," in *2017 International Conference on Systems, Signals and Image Processing (IWSSIP)*. IEEE, May 2017, pp. 1–9.

[8] J. M. Boyce, R. Dore, A. Dziembowski, J. Fleureau, J. Jung, B. Kroon, B. Salahieh, V. K. M. Vadakital, and L. Yu, "MPEG Immersive Video Coding Standard," *Proceedings of the IEEE*, vol. 109, no. 9, pp. 1521–1536, Sep. 2021.

[9] S. Schwarz, M. Preda, V. Baroncini, M. Budagavi, P. Cesar, P. A. Chou, R. A. Cohen, M. Krivokuća, S. Lasserre, Z. Li, J. Llach, K. Mammou, R. Mekuria, O. Nakagami, E. Siahaan, A. Tabatabai, A. M. Tourapis, and V. Zakharchenko, "Emerging MPEG Standards for Point Cloud Compression," *IEEE Journal on Emerging and Selected Topics in Circuits and Systems*, vol. 9, no. 1, pp. 133–148, 2019.

[10] D. Graziosi, O. Nakagami, S. Kuma, A. Zaghetto, T. Suzuki, and A. Tabatabai, "An overview of ongoing point cloud compression standardization activities: video-based (V-PCC) and

geometry-based (G-PCC)," *APSIPA Transactions on Signal and Information Processing*, vol. 9, 2020.

[11] W. Zou, S. Zhang, F. Yang, and M. Preda, "Standardization Status of MPEG Video-based Dynamic Mesh Coding (V-DMC)," in *ICASSP 2025*. IEEE, Apr. 2025, pp. 1–5.

[12] T. M. Borges, "Fractional Super-Resolution of Voxelized Point Clouds," Master's thesis, Departamento de Engenharia Elétrica, Universidade de Brasília, Brasília, DF, Brazil, Jan. 2021, PPGENE.DM-759/20.

[13] T. M. Borges, D. C. Garcia, and R. L. de Queiroz, "Fractional Super-Resolution of Voxelized Point Clouds," *IEEE Trans. on Image Processing*, vol. 31, pp. 1380–1390, 2022.

[14] T. M. Borges, R. U. B. Ferreira, D. C. Garcia, and R. L. de Queiroz, "Using Fractional Super-Resolution to Improve Lossy Compression of Point Cloud Geometry," *Journal of Communication and Information Systems*, vol. 38, pp. 169–173, 2023.

[15] R. U. B. Ferreira, T. M. Borges, D. C. Garcia, and R. L. de Queiroz, "[G-PCC] Post-processing geometry super-resolution based on lookup tables," ISO/IEC JTC 1/SC29/WG7, Online, input doc. m58216, Oct. 2021.

[16] T. M. Borges, R. U. B. Ferreira, D. C. Garcia, and R. L. de Queiroz, "[G-PCC][EE13.56] Crosscheck report on Point Cloud Geometry Compression with a Hierarchical Prior," ISO/IEC JTC 1/SC29/WG7, Online, input doc. m59361, Apr. 2022.

[17] R. U. B. Ferreira, T. M. Borges, D. C. Garcia, and R. L. de Queiroz, "[G-PCC][EE13.56.a] Report on GPCC post-processing geometry super-resolution," ISO/IEC JTC 1/SC29/WG7, Mainz, DE, input doc. m60986, Oct. 2022.

[18] M. Krivokuća, T. M. Borges, and R. L. de Queiroz, "Zerotree Coding of Subdivision Wavelet Coefficients in Dynamic Time-Varying Meshes," *IEEE Transactions on Image Processing*, vol. 34, pp. 1810–1819, 2025.

[19] M. Krivokuća, T. Borges, O. Mocquard, J.-E. Marvie, O. Roupin, and J. Ricard, "Progressive Encoding and Adaptive Reconstruction of Dynamic Meshes Using Zerotree Coding," WO International Patent WO/2024/061 661, 2023. [Online]. Available: https://patentscope.wipo.int/search/en/detail.jsf?docId=WO2024061661

[20] M. Krivokuća, T. M. Borges, O. Roupin, and G. Sandri, "[V-DMC][EE4.7-related] Hierarchical Zerotree Coding of Subdivision Wavelet Coefficients," ISO/IEC JTC 1/SC29/WG7, Antalya, TR, input doc m63183, Apr. 2023.

[21] M. Krivokuća, T. M. Borges, and O. Roupin, "[V-DMC][EE4.7-related] Zerotree Coding Syntax for m63183," ISO/IEC JTC 1/SC29/WG7, Antalya, TR, input doc m63186, Apr. 2023.

[22] M. Krivokuća and T. M. Borges, "[V-DMC][EE4.7] Updated Zerotree Results for Wavelet Coefficients Coding," ISO/IEC JTC 1/SC29/WG7, Geneva, CZ, input doc m63186, Jun. 2023.

[23] ——, "[V-DMC] Clarifications on Zerotree Displacement Coding," ISO/IEC JTC 1/SC29/WG7, Hannover, DE, input doc m63186, Oct. 2023.

[24] A. B. Tucker, Ed., *Computer science handbook*, 2nd ed. Chapman & Hall/CRC, 2004.

[25] A. C. Telea, *Data Visualization*, 2nd ed. Taylor & Francis Ltd., 2014.

[26] M. Levoy. (2014) The Stanford 3D Scanning Repository. Stanford University Computer Graphics Laboratory. Accessed: 23-04-2025. [Online]. Available: http://graphics.stanford.edu/data/3Dscanrep/

[27] G. Turk and M. Levoy, "Zippered polygon meshes from range images," in *Proceedings of the 21st annual conference on Computer graphics and interactive techniques - SIGGRAPH '94*. ACM Press, 1994.

[28] M. Zwicker, H. Pfister, J. van Baar, and M. Gross, "Surface splatting," in *Proceedings of the 28th annual conference on Computer graphics and interactive techniques - SIGGRAPH '01*, ser. SIGGRAPH '01. New York, NY, USA: ACM Press, 2001, pp. 371–378.

[29] J.-E. Marvie, M. Krivokuća, and D. Graziosi, "Coding of Dynamic 3D Meshes," in *Immersive Video Technologies*, G. Valenzise, M. Alain, E. Zerman, and C. Ozcinar, Eds. Elsevier Inc., 2023, ch. 14, pp. 387–423.

[30] M. Krivokuća, "Progressive Compression of 3D Mesh Geometry Using Sparse Approximations from Redundant Frame Dictionaries," phdthesis, Department of Electrical and Computer Engineering, The University of Auckland, New Zealand, 2015.

[31] G. J. Sullivan, J.-R. Ohm, W.-J. Han, and T. Wiegand, "Overview of the High Efficiency Video Coding (HEVC) Standard," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 22, no. 12, pp. 1649–1668, Dec. 2012.

[32] B. Bross, Y.-K. Wang, Y. Ye, S. Liu, J. Chen, G. J. Sullivan, and J.-R. Ohm, "Overview of the Versatile Video Coding (VVC) Standard and its Applications," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 31, pp. 3736–3764, Oct. 2021.

[33] R. Mukundan, *3D Mesh Processing and Character Animation*, ser. Springer eBook Collection. Cham: Springer, 2022.

[34] A. J. Parker, "Stereoscopic Vision," in *Encyclopedia of Neuroscience*. Elsevier, 2009, pp. 411–417.

[35] Wikipedia contributors. (2020, Sep.) Depth perception. Online. Wikipedia, The Free Encyclopedia. Accessed: 23-04-2025. [Online]. Available: https://en.wikipedia.org/wiki/Depth_perception

[36] R. S. Laramee. (2019) Data Visualization Classes 2019. Computer Science Department at Swansea University. Accessed on: 23-04-2025. [Online]. Available: https://www.youtube.com/playlist?list=PLZo40sVmw_4No1_K-Xllf04O977PYngq1

[37] W. E. Lorensen and H. E. Cline, "Marching cubes: A high resolution 3D surface construction algorithm," *ACM SIGGRAPH Computer Graphics*, vol. 21, no. 4, pp. 163–169, Aug. 1987.

[38] T. S. Newman and H. Yi, "A survey of the marching cubes algorithm," *Computers & Graphics*, vol. 30, no. 5, pp. 854–879, Oct. 2006.

[39] M. Scholz, J. Bender, and C. Dachsbacher, "Real-Time Isosurface Extraction With View-Dependent Level of Detail and Applications," *Computer Graphics Forum*, vol. 34, no. 1, pp. 103–115, Oct. 2014.

[40] K. J. Büscher, J. P. Degel, and J. Oellerich, "A Comprehensive Survey of Isocontouring Methods: Applications, Limitations and Perspectives," *MDPI Algorithms*, vol. 17, no. 2, p. 83, Feb. 2024.

[41] V. Castelli and L. D. Bergman, Eds., *Image Databases*. John Wiley & Sons, 2001.

[42] R. Szeliski, *Computer Vision*. Springer London, 2011.

[43] R. Gonzalez, *Digital image processing*, 3rd ed. Upper Saddle River, N.J: Prentice Hall, 2008.

[44] ITU-R BT.709-6, "Parameter values for the HDTV standards for production and international programme exchange," International Telecommunication Union, Recommendation, May 2015.

[45] K. Sayood, *Introduction to data compression*, 5th ed. Cambridge: Morgan Kaufmann Publishers, 2018.

[46] T. M. Cover, *Elements of information theory*, 2nd ed., J. A. Thomas, Ed. Hoboken, N.J: Wiley-Interscience, 2006, includes bibliographical references (pages 689-721) and index.

[47] Y. Shoham and A. Gersho, "Efficient bit allocation for an arbitrary set of quantizers (speech coding)," *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. 36, no. 9, pp. 1445–1453, 1988.

[48] A. Ortega and K. Ramchandran, "Rate-distortion methods for image and video compression," *IEEE Signal Processing Magazine*, vol. 15, no. 6, pp. 23–50, 1998.

[49] G. J. Sullivan and T. Wiegand, "Rate-distortion optimization for video compression," *IEEE Signal Processing Magazine*, vol. 15, no. 6, pp. 74–90, 1998.

[50] G. Bjøntegaard, "Calculation of average PSNR differences between RD-curves," *VCEG-M33*, 2001.

[51] C. Herglotz, H. Och, A. Meyer, G. Ramasubbu, L. Eichermüler, M. Kräzler, F. Brand, K. Fischer, D. T. Nguyen, A. Regensky, and A. Kaup, "The Bjøtegaard Bible Why Your Way of Comparing Video Codecs May Be Wrong," *IEEE Transactions on Image Processing*, vol. 33, pp. 987–1001, 2024.

[52] R. L. de Queiroz, D. C. Garcia, Y.-H. Chen, R. Conceição, and W.-H. Peng, "Application Space and the Rate-Distortion-Complexity Analysis of Neural Video CODECs," *Preprint*, 2025.

[53] E. Alexiou, I. Viola, T. M. Borges, T. A. Fonseca, R. L. de Queiroz, and T. Ebrahimi, "A comprehensive study of the rate-distortion performance in MPEG point cloud compression," *APSIPA Transactions on Signal and Information Processing*, vol. 8, 2019.

[54] A. Javaheri, C. Brites, F. Pereira, and J. Ascenso, "Point Cloud Rendering After Coding: Impacts on Subjective and Objective Quality," *IEEE Transactions on Multimedia*, vol. 23, pp. 4049–4064, 2021.

[55] D. Tian, H. Ochimizu, C. Feng, R. Cohen, and A. Vetro, "Updates and Integration of Evaluation Metric Software for PCC," ISO/IEC MPEG JTC1/SC29/WG11, Hobart, AU, Input Document m40522, Apr. 2017, 2017.

[56] J.-E. Marvie, Y. Nehmé, D. Graziosi, and G. Lavoué, "Crafting the MPEG metrics for objective and perceptual quality assessment of volumetric videos," *Qual. User Exp.*, vol. 8, no. 1, Jun. 2023.

[57] R. L. de Queiroz and P. A. Chou, "Motion-Compensated Compression of Dynamic Voxelized Point Clouds," *IEEE Transactions on Image Processing*, vol. 26, no. 8, pp. 3886–3895, Aug. 2017.

[58] Wikipedia contributors. (2020, Oct.) Hausdorff distance. Wikipedia, The Free Encyclopedia. Accessed: 23-04-2025. [Online]. Available: https://en.wikipedia.org/wiki/Hausdorff_distance

[59] D. Girardeau-Montaut, M. Roux, R. Marc, and G. Thibault, "Change Detection on Points Cloud Data Acquired with a Ground Laser Scanner," in *Proceedings of the ISPRS Workshop Laser scanning 2005*, G. Vosselman and C. Brenner, Eds., vol. XXXVI-3/W19. Enschede, the Netherlands: International Society for Photogrammetry and Remote Sensing, Sep. 2005, pp. 30–35.

[60] D. Tian, H. Ochimizu, C. Feng, R. Cohen, and A. Vetro, "Evaluation Metrics for Point Cloud Compression," ISO/IEC MPEG JTC1/SC29/WG11, Chengdu, China, Input Document m39316, Oct. 2016.

[61] D. Tian, H. Ochimizu, C. Feng, R. Cohen, and A. Vetro, "Geometric distortion metrics for point cloud compression," in *2017 IEEE International Conference on Image Processing (ICIP)*, Sep. 2017, pp. 3460–3464.

[62] H. Hoppe, T. DeRose, T. Duchamp, J. McDonald, and W. Stuetzle, "Surface Reconstruction from Unorganized Points," *SIGGRAPH Comput. Graph.*, vol. 26, no. 2, pp. 71–78, Jul. 1992.

[63] MPEG 3DG, "Common test conditions for point cloud compression," ISO/IEC JTC1/SC29/WG11, Gothenburg, SE, Approved WG 11 document N18883, Jul. 2019.

[64] E. Alexiou and T. Ebrahimi, "Exploiting user interactivity in quality assessment of point cloud imaging," in *2019 Eleventh International Conference on Quality of Multimedia Experience (QoMEX)*. IEEE, Jun. 2019.

[65] T.-Y. Kuo, Y.-J. Wei, and K.-H. Wan, "Color Image Quality Assessment Based on VIF," in *2019 3rd International Conference on Imaging, Signal Processing and Communication (ICISPC)*. IEEE, Jul. 2019.

[66] Z. Wang, A. C. Bovik, H. R. Sheikh, and E. P. S. Simoncelli, "Image Quality Assessment: From Error Visibility to Structural Similarity," *IEEE Transactions on Image Processing*, vol. 13, no. 4, pp. 600–612, Apr. 2004.

[67] LIVE – Laboratory for Image & Video Engineering. Image & Video Quality Assessment Algorithms. The University of Texas at Austin. Accessed on: 23-04-2025. [Online]. Available: https://live.ece.utexas.edu/research/Quality/index_algorithms.htm

[68] H. R. Sheikh and A. C. Bovik, "Image information and visual quality," *IEEE Transactions on Image Processing*, vol. 15, no. 2, pp. 430–444, Feb. 2006.

[69] H. R. Sheikh and A. C. Bovik. Image Information and Visual Quality. The University of Texas at Austin. Accessed: 23-04-2025. [Online]. Available: https://live.ece.utexas.edu/research/Quality/VIF.htm

[70] ISO/IEC JTC1/SC29. (2020, Jun.) Future of SC 29 with JPEG and MPEG. Online. ISO/IEC JTC1/SC29. Accessed: 28-05-2025. [Online]. Available: https://jtc1info.org/future-of-sc-29-with-jpeg-and-mpeg/

[71] J. Ozer. (2020, Jul.) MPEG: What Happened? Online. Streaming Media. Accessed: 28-05-2025. [Online]. Available: https://www.streamingmedia.com/Articles/ReadArticle.aspx?ArticleID=141678

[72] MPEG 3DG, "MPEG 3DG and Requirements: Call for proposals for point cloud compression v2," ISO/IEC JTC1/SC29/WG11, Hobart, AU, Approved WG 11 document N16763, Apr. 2017.

[73] R. Mekuria, K. Blom, and P. Cesar, "Design, Implementation, and Evaluation of a Point Cloud Codec for Tele-Immersive Video," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 27, no. 4, pp. 828–842, Apr. 2017.

[74] L. Cui, R. Mekuria, M. Preda, and E. S. Jang, "Point-Cloud Compression: Moving Picture Experts Group's New Standard in 2020," *IEEE Consumer Electronics Magazine*, vol. 8, no. 4, pp. 17–21, Jul. 2019.

[75] MPEG 3DG, "Report on PCC CfP answers," ISO/IEC JTC1/SC29/WG11, Macau, CN, Approved WG 11 document w17251, Oct. 2017.

[76] ISO/IEC JTC 1/SC 29, *ISO/IEC 23090-5:2021 - Information technology – Coded representation of immersive media – Part 5: Visual volumetric video-based coding (V3C) and video-based point cloud compression (V-PCC)*, 1st ed., 2021.

[77] L. Ilola, L. Kondrad, S. Schwarz, and A. Hamza, "An Overview of the MPEG Standard for Storage and Transport of Visual Volumetric Video-Based Coding," *Frontiers in Signal Processing*, vol. 2, Apr. 2022.

[78] MPEG WG7 3DGH, "CfP for Dynamic Mesh Coding," ISO/IEC JTC 1/SC 29/WG 7, Online, Tech. Rep. w21000/N00231, Oct. 2021.

[79] D. B. Graziosi, S. Kuma, K. Hayashi, O. Nakagami, and A. Tabatabai, "[V-CG] Study of Dynamic Mesh Coding CfP Submission Proposals," ISO/IEC JTC1/SC29/WG7, Online, Tech. Rep. m59625, Apr. 2022.

[80] K. Mammou, J. Kim, A. M. Tourapis, D. Podborski, and D. Flynn, "Video and Subdivision based Mesh Coding," in *EUVIP2022*. Lisbon, Portugal: IEEE, 2022, pp. 1–6.

[81] MPEG Technical Requirements, "Updated Call for Proposals for AI-based Point Cloud Coding," ISO/IEC JTC1/SC29/WG2, Sapporo, JP, Output document N396/w24310, Jul. 2024.

[82] MPEG WG7 3DGH, "[GSC][JEE6.4] Draft use cases and requirements for Gaussian splat coding," ISO/IEC JTC1/SC29/WG7, Geneva, CH, input doc m71640, Jan. 2025.

[83] M. T. Bagdasarian, P. Knoll, Y. Li, F. Barthel, A. Hilsmann, P. Eisert, and W. Morgenstern, "3DGS.zip: A survey on 3D Gaussian Splatting Compression Methods," *Computer Graphics Forum*, Apr. 2025.

[84] B. Kerbl, G. Kopanas, T. Leimkühler, and G. Drettakis, "3D Gaussian Splatting for Real-Time Radiance Field Rendering," *ACM Transactions on Graphics*, vol. 42, no. 4, July 2023.

[85] G. Valenzise, M. Quach, D. Tian, J. Pang, and F. Dufaux, "Point Cloud Compression," in *Immersive Video Technologies*, G. Valenzise, M. Alain, E. Zerman, and C. Ozcinar, Eds. Elsevier, 2023, ch. 13, pp. 357–385.

[86] D. Marpe, H. Schwarz, and T. Wiegand, "Context-based adaptive binary arithmetic coding in the H.264/AVC video compression standard," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 13, no. 7, pp. 620–636, Jul. 2003.

[87] W. Zhang, F. Yang, Y. Xu, and M. Preda, "Standardization Status of MPEG Geometry-Based Point Cloud Compression (G-PCC) Edition 2," in *2024 Picture Coding Symposium (PCS)*. IEEE, Jun. 2024, pp. 1–5.

[88] MPEG 3DG, "G-PCC codec description v5," ISO/IEC JTC1/SC29/WG11, Geneva, CH, Approved WG 11 document w18891/N18891, Oct. 2019.

[89] D. Meagher, "Geometric modeling using octree encoding," *Computer Graphics and Image Processing*, vol. 19, no. 2, pp. 129–147, Jul. 1982.

[90] R. L. de Queiroz and P. A. Chou, "Compression of 3D Point Clouds Using a Region-Adaptive Hierarchical Transform," *IEEE Transactions on Image Processing*, vol. 25, no. 8, pp. 3947–3956, Aug. 2016.

[91] R. L. de Queiroz, D. C. Garcia, P. A. Chou, and D. A. Florencio, "Distance-based probability model for octree coding," *IEEE Signal Processing Letters*, vol. 25, 2018.

[92] S. Lasserre and D. Flynn, "[PCC] Intra mode for geometry coding in TMC3," ISO/IEC MPEG JTC1/SC29/WG11, Ljubljana, Slovenia, Input contribution m43600, Jul. 2018.

[93] S. Lasserre, D. Flynn, and S. Qu, "[PCC] An overview of OBUF and neighbour usage for geometry coding," ISO/IEC MPEG JTC1/SC29/WG11, Marrakesh, Morocco, Input contribution m45811, Jan. 2019.

[94] S. Lasserre and D. Flynn, "[PCC] Inference of a mode using point location direct coding in TMC3," ISO/IEC MPEG JTC1/SC29/WG11, Gwangju, Korea, Input contribution m42239, Jan. 2018.

[95] E. Pavez, P. A. Chou, R. L. de Queiroz, and A. Ortega, "Dynamic polygon clouds: representation and compression for VR/AR," *APSIPA Trans. Signal and Inf. Process.*, vol. 7, 2018.

[96] P. A. Chou, "Trisoup C++ reference code for TMC13," ISO/IEC MPEG JTC1/SC29/WG11, Ljubjana, Slovenia, Input Document m43786, Jul. 2018.

[97] MPEG 3DG, "PCC Test Model Category 3 v0," ISO/IEC JTC1/SC29/WG11, Macau, China, Approved WG 11 document w17249/N17249, Oct. 2017.

[98] K. Mammou, A. Tourapis, J. Kim, F. Robinet, V. Valentin, and Y. Su, "Lifting Scheme for Lossy Attribute Encoding in TMC1," ISO/IEC MPEG JTC1/SC29/WG11, San Diego, US, Input contribution m42640, Apr. 2018.

[99] G. P. Sandri, P. A. Chou, M. Krivokuća, and R. L. de Queiroz, "Integer Alternative for the Region-Adaptive Hierarchical Transform," *IEEE Signal Processing Letters*, vol. 26, no. 9, pp. 1369–1372, Sep. 2019.

[100] G. Sandri, R. L. de Queiroz, and P. A. Chou, "Comments on "Compression of 3D Point Clouds Using a Region-Adaptive Hierarchical Transform"," 2018.

[101] S. Lasserre and D. Flynn, "[G-PCC][new proposal] On an improvement of RAHT to exploit attribute correlation," ISO/IEC MPEG JTC1/SC29/WG11, Geneva, CH, Tech. Rep. m47378, Mar. 2019.

[102] ——, "G-PCC CE13.18 report on upsampled transform domain prediction in RAHT," ISO/IEC MPEG JTC1/SC29/WG11, Gothenburg, Sweden, Input document m49380, Jul. 2019.

[103] W. Sweldens, "The Lifting Scheme: A Custom-Design Construction of Biorthogonal Wavelets," *Applied and Computational Harmonic Analysis*, vol. 3, no. 2, pp. 186–200, Apr. 1996.

[104] K. Mammou, N. Stefanoski, H. Kirchhoffer, K. Muller, T. Zaharia, F. Preteux, D. Marpe, and J. Ostermann, "The New MPEG-4/FAMC Standard for Animated 3D Mesh Compression," in *2008 3DTV Conference: The True Vision - Capture, Transmission and Display of 3D Video*. IEEE, May 2008, pp. 97–100.

[105] ISO/IEC JTC 1/SC 29, *ISO/IEC 14496-16:2006/Amd 2:2009 - Information technology – Coding of audio-visual objects – Part 16: Animation Framework eXtension (AFX) – Amendment 2: Frame-based Animated Mesh Compression (FAMC)*, 2nd ed., 2009.

[106] ——, *ISO/IEC 14496-16:2011 - Information technology – Coding of audio-visual objects – Part 16: Animation Framework eXtension (AFX)*, 4th ed., 2011.

[107] T. Matsuyama, X. Wu, T. Takai, and T. Wada, "Real-Time Dynamic 3-D Object Shape Reconstruction and High-Fidelity Texture Mapping for 3-D Video," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 14, no. 3, pp. 357–369, Mar. 2004.

[108] J. Starck and A. Hilton, "Surface Capture for Performance-Based Animation," *IEEE Computer Graphics and Applications*, vol. 27, no. 3, pp. 21–31, May 2007.

[109] H. Habe, Y. Katsura, and T. Matsuyama, "Skin-off: representation and compression scheme for 3D video," in *Picture Coding Symposium*, 2004, pp. 301–306.

[110] S.-R. Han, T. Yamasaki, and K. Aizawa, "3D video compression based on extended block matching algorithm," in *IEEE Int. Conf. on Image Processing (ICIP)*. IEEE, 2006, pp. 525–528.

[111] ——, "Time-varying mesh compression using an extended block matching algorithm," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 17, no. 11, pp. 1506–1518, 2007.

[112] ——, "Geometry compression for time-varying meshes using coarse and fine levels of quantization and run-length encoding," in *IEEE Int. Conf. on Image Processing (ICIP)*, 2008, pp. 1045–1048.

[113] T. Yamasaki and K. Aizawa, "Patch-based compression for time-varying meshes," in *IEEE Int. Conf. on Image Processing (ICIP)*. IEEE, 2010, pp. 3433–3436.

[114] L. Yamasaki and K. Aizawa, "Bit allocation of vertices and colors for patch-based coding in time-varying meshes," in *28th Picture Coding Symposium*, 2010, pp. 162–165.

[115] K. Mammou, J. Kim, A. Tourapis, D. Podborski, and K. Kolarov, "[V-CG] Apple's Dynamic Mesh Coding CfP Response," ISO/IEC JTC 1/SC 29/WG 7, Online, Tech. Rep. m59281, Apr. 2022.

[116] J.-E. Marvie, C. Guede, J. Ricard, O. Mocquard, M. Krivokuća, and F.-L. Tariolle, "[V-CG] InterDigital's Response to Dynamic Mesh Coding CfP," ISO/IEC JTC 1/SC 29/WG 7, Online, Tech. Rep. m59285, Apr. 2022.

[117] J.-E. Marvie, M. Krivokuca, C. Guede, J. Ricard, O. Mocquard, and F.-L. Tariolle, "Compression of Time-Varying Textured Meshes using Patch Tiling and Image-based Tracking," in *2022 10th European Workshop on Visual Information Processing (EUVIP)*. IEEE, Sep. 2022, pp. 1–6.

[118] P. R. Alface, A. Martemianov, S. Schwarz, L. I. abd Lukasz Kondrad, J. Szabo, and C. Bachhuber, "[V-CG] Nokia's response to CfP for Dynamic Mesh compression," ISO/IEC JTC 1/SC 29/WG 7, Online, Tech. Rep. m59274, Apr. 2022.

[119] P. R. Alface, A. Martemianov, L. Ilola, L. Kondrad, C. Bachhuber, and S. Schwarz, "V3C-based Coding of Dynamic Meshes," in *2022 10th European Workshop on Visual Information Processing (EUVIP)*. IEEE, Sep. 2022, pp. 1–6.

[120] X. Z. andChao Huang, J. Tian, X. Xu, and S. Liu, "[V-CG] Tencent's Dynamic Mesh Coding CfP response," ISO/IEC JTC 1/SC 29/WG 7, Online, Tech. Rep. m59295, Apr. 2022.

[121] C. Huang, X. Zhang, J. Tian, X. Xu, and S. Liu, "Boundary-Preserved Geometry Video for Dynamic Mesh Coding," in *2022 Picture Coding Symposium (PCS)*, vol. 34. IEEE, Dec. 2022, pp. 133–137.

[122] D. B. Graziosi, "Video-Based Dynamic Mesh Coding," in *IEEE Int. Conf. on Image Processing (ICIP)*. IEEE, 2021, pp. 3133–3137.

[123] D. B. Graziosi, S. Kuma, K. Hayashi, O. Nakagami, and A. Tabatabai, "[V-CG] Sony's Dynamic Mesh Coding Call for Proposal Response," ISO/IEC JTC 1/SC 29/WG 7, Online, Tech. Rep. m59284, Apr. 2022.

[124] M. Sabin, "Recent Progress in Subdivision: a Survey," in *Advances in Multiresolution for Geometric Modelling*, ser. Mathematics and Visualization, N. A. Dodgson, M. S. Floater, and M. A. Sabin, Eds. Berlin, Heidelberg: Springer, 2005, pp. 203–230.

[125] A. Lee, H. Moreton, and H. Hoppe, "Displaced subdivision surfaces," in *Proceedings of the 27th annual conference on Computer graphics and interactive techniques - SIGGRAPH '00*, ser. SIGGRAPH '00. ACM Press, 2000, pp. 85–94.

[126] A. Maggiordomo, H. Moreton, and M. Tarini, "Micro-mesh construction," *ACM Transactions on Graphics*, vol. 42, no. 4, pp. 1–18, Jul. 2023.

[127] J.-E. Marvie and O. Mocquard, "[V-DMC][EE4.4] Edgebreaker base mesh codec integrationAuthor: Jean-Eudes Marvie, Olivier Mocquard," ISO/IEC JTC 1/SC 29/WG 7, Antalya, TR, Tech. Rep. m62603, Apr. 2023.

[128] J. Rossignac, "Edgebreaker: connectivity compression for triangle meshes," *IEEE Trans. Vis. Comput. Graphics*, vol. 5, pp. 47–61, 1999.

[129] K. Kolarov and W. Lynch, "Compression of functions defined on surfaces of 3D objects," in *Proceedings DCC'97. Data Compression Conference*. IEEE, 1997, pp. 281–290.

[130] C. Huang, X. Zhang, X. Xu, J. Tian, and S. Liu, "Arithmetic Coding of Displacements for Subdivision-based Mesh Compression," ISO/IEC JTC1/SC29/WG7, Online, Tech. Rep. m60300, Jul. 2022.

[131] W. Zou, H. Huang, F. Yang, N. Wang, and Z. Lv, "[V-DMC][EE4.4-related] On supporting duplicate vertices merging and higher completeness for lossless MPEG Edgebreaker," ISO/IEC JTC1/SC29/WG7, Geneva, Tech. Rep. m64206, Jul. 2023.

[132] M. Garland and P. S. Heckbert, "Surface simplification using quadric error metrics," in *SIGGRAPH*, USA, Aug. 1997, pp. 209–216.

[133] X. Huang and C. Walbourn. (2025) UVAtlas - isochart texture atlasing. Microsoft Research China. Accessed: 10-08-2025. [Online]. Available: https://github.com/Microsoft/UVAtlas

[134] D. B. Graziosi and K. Hayashi, "Dynamic Mesh Coding Using Orthogonal Atlas Projection," in *2024 Picture Coding Symposium (PCS)*. IEEE, Jun. 2024, pp. 1–5.

[135] H. Hoppe, T. DeRose, T. Duchamp, M. Halstead, H. Jin, J. McDonald, J. Schweitzer, and W. Stuetzle, "Piecewise smooth surface reconstruction," in *Proceedings of the 21st annual conference on Computer graphics and interactive techniques - SIGGRAPH '94*, ser. SIGGRAPH '94. ACM Press, 1994, pp. 295–302.

[136] C. Loop, "Smooth Subdivision Surfaces Based on Triangles," Master's thesis, Department of Mathematics, University of Utah, Jan. 1987.

[137] M. Lounsbery, T. D. DeRose, and J. Warren, "Multiresolution analysis for surfaces of arbitrary topological type," *ACM TOG*, vol. 16, no. 1, pp. 34–73, 1997.

[138] W. Sweldens, "The Lifting Scheme: A Construction of Second Generation Wavelets," *SIAM J. Math. Anal.*, vol. 29, no. 2, pp. 511–546, 1998.

[139] M. Kraus, "The pull-push algorithm revisited," in *Proceedings GRAPP 2009*, 2009.

[140] K. Mammou, J. Kim, V. Valentin, F. Robinet, A. Tourapis, and Y. Su, "CE2.12 related: Sparse Linear model based padding method for the texture images," ISO/IEC MPEG JTC1/SC29/WG11, Macau, CH, Input contribution m44837, Oct. 2018.

[141] D. C. Garcia, A. L. Souto, G. P. Sandri, T. M. Borges, and R. L. de Queiroz, "Point Cloud Reconstruction From Truncated Geometry-Based Streams," *Journal of Communication and Information Systems*, vol. 38, no. 1, pp. 77–84, 2023.

[142] H. Schwarz, D. Marpe, and T. Wiegand, "Overview of the Scalable Video Coding Extension of the H.264/AVC Standard," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 17, no. 9, pp. 1103–1120, Sep. 2007.

[143] J. M. Boyce, Y. Ye, J. Chen, and A. K. Ramasubramonian, "Overview of SHVC: Scalable Extensions of the High Efficiency Video Coding Standard," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 26, no. 1, pp. 20–34, Jan. 2016.

[144] Y.-K. Wang, R. Skupin, M. M. Hannuksela, S. Deshpande, Hendry, V. Drugeon, R. Sjoberg, B. Choi, V. Seregin, Y. Sanchez, J. M. Boyce, W. Wan, and G. J. Sullivan, "The High-Level Syntax of the Versatile Video Coding (VVC) Standard," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 31, no. 10, pp. 3779–3800, Oct. 2021.

[145] J. M. Shapiro, "Embedded image coding using zerotrees of wavelet coefficients," *IEEE Trans. Signal Process.*, vol. 41, no. 12, pp. 3445–3462, 1993.

[146] A. Said and W. A. Pearlman, "A new, fast, and efficient image codec based on set partitioning in hierarchical trees," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 6, no. 3, pp. 243–250, 1996.

[147] A. Skodras, C. Christopoulos, and T. Ebrahimi, "The JPEG 2000 still image compression standard," *IEEE Signal Processing Magazine*, vol. 18, no. 5, pp. 36–58, 2001.

[148] D. Taubman, "High performance scalable image compression with EBCOT," *IEEE Transactions on Image Processing*, vol. 9, no. 7, pp. 1158–1170, Jul. 2000.

[149] A. L. Souto, V. F. Figueiredo, P. A. Chou, and R. L. de Queiroz, "Set Partitioning in Hierarchical Trees for Point Cloud Attribute Compression," *IEEE Signal Processing Letters*, vol. 28, pp. 1903–1907, 2021.

[150] V. F. Figueiredo and R. L. de Queiroz, "Embedded Bit-Stream Region-of-Interest Coding of Point Cloud Attributes," in *2024 IEEE 26th International Workshop on Multimedia Signal Processing (MMSP)*. IEEE, Oct. 2024, pp. 1–6.

[151] D. Huffman, "A Method for the Construction of Minimum-Redundancy Codes," *Proceedings of the IRE*, vol. 40, no. 9, pp. 1098–1101, Sep. 1952.

[152] G. G. Langdon, "An introduction to arithmetic coding," *IBM Journal of Research and Development*, vol. 28, no. 2, pp. 135–149, Mar. 1984.

[153] W. A. Pearlman and A. Said, *Digital Signal Compression: Principles and Practice*. Cambridge University Press, Oct. 2011.

[154] D. C. Garcia, T. A. Fonseca, R. U. Ferreira, and R. L. de Queiroz, "Geometry Coding for Dynamic Voxelized Point Clouds Using Octrees and Multiple Contexts," *IEEE Transactions on Image Processing*, vol. 29, pp. 313–322, 2019.

[155] M. Alexa, J. Behr, D. Cohen-Or, S. Fleishman, D. Levin, and C. T. Silva, "Computing and rendering point set surfaces," *IEEE Transactions on Visualization and Computer Graphics*, vol. 9, no. 1, pp. 3–15, 2003.

[156] A. Okabe, B. Boots, K. Sugihara, and S. N. Chiu, *Spatial Tessellations: Concepts and Applications of Voronoi Diagrams*. John Wiley & Sons, Inc., Jul. 2000.

[157] G. Guennebaud and M. Gross, "Algebraic point set surfaces," *ACM Trans. Graph.*, vol. 26, no. 3, p. 23, Jul. 2007.

[158] A. C. Öztireli, G. Guennebaud, and M. Gross, "Feature Preserving Point Set Surfaces based on Non-Linear Kernel Regression," *Computer Graphics Forum*, vol. 28, no. 2, pp. 493–501, Apr. 2009.

[159] H. Huang, S. Wu, M. Gong, D. Cohen-Or, U. Ascher, and H. R. Zhang, "Edge-Aware Point Set Resampling," *ACM Trans. Graph.*, vol. 32, no. 1, Feb. 2013.

[160] A. Hamdi-Cherif, J. Digne, and R. Chaine, "Super-Resolution of Point Set Surfaces Using Local Similarities," *Computer Graphics Forum*, vol. 37, no. 1, pp. 60–70, Jun. 2017.

[161] C. Dinesh, G. Cheung, and I. V. Bajić, "3D Point Cloud Super-Resolution via Graph Total Variation on Surface Normals," in *IEEE International Conference on Image Processing (ICIP)*. IEEE, Sep. 2019.

[162] ——, "Super-Resolution of 3D Color Point Clouds Via Fast Graph Total Variation," in *ICASSP 2020 - 2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2020, pp. 1983–1987.

[163] S. Dinas and H. J. Martínez, *Delaunay Triangulation*. Springer International Publishing, 2020, pp. 1–6.

[164] L. Yu, X. Li, C.-W. Fu, D. Cohen-Or, and P.-A. Heng, "PU-Net: Point Cloud Upsampling Network," in *IEEE/CVF Conf. on Comput. Vision and Pattern Recog. (CVPR)*. IEEE, Jun. 2018.

[165] ——, "EC-Net: An Edge-Aware Point Set Consolidation Network," in *European Conf. on Computer Vision (ECCV)*. Springer Intl. Publishing, 2018, pp. 398–414.

[166] W. Yifan, S. Wu, H. Huang, D. Cohen-Or, and O. Sorkine-Hornung, "Patch-Based Progressive 3D Point Set Upsampling," in *IEEE/CVF Conf. on Comput. Vision and Pattern Recog. (CVPR)*. IEEE, Jun. 2019.

[167] R. Li, X. Li, C.-W. Fu, D. Cohen-Or, and P.-A. Heng, "PU-GAN: a Point Cloud Upsampling Adversarial Network," in *IEEE Intl. Conf. on Computer Vision (ICCV)*, Oct. 2019.

[168] H. Wu, J. Zhang, and K. Huang, "Point Cloud Super Resolution with Adversarial Residual Graph Networks," *British Mach. Vis. Conf. (BMVC)*, 2020.

[169] G. Qian, A. Abualshour, G. Li, A. Thabet, and B. Ghanem, "PU-GCN: Point Cloud Upsampling using Graph Convolutional Networks," in *IEEE/CVF Conf. on Comput. Vision and Pattern Recog. (CVPR)*, 2021.

[170] Y. Qian, J. Hou, S. Kwong, and Y. He, "PUGeo-Net: A Geometry-Centric Network for 3D Point Cloud Upsampling," in *Computer Vision – ECCV 2020*. Springer Intl. Publishing, 2020, pp. 752–769.

[171] S. Ye, D. Chen, S. Han, Z. Wan, and J. Liao, "Meta-PU: An Arbitrary-Scale Upsampling Network for Point Cloud," *IEEE Trans. Vis. Comput. Graphics*, pp. 1–1, 2021.

[172] D. C. Garcia, T. A. Fonseca, and R. L. de Queiroz, "Example-Based Super-Resolution for Point-Cloud Video," in *IEEE International Conference on Image Processing (ICIP)*. IEEE, 2018, pp. 2959–2963.

[173] M. Corsini, P. Cignoni, and R. Scopigno, "Efficient and Flexible Sampling with Blue Noise Properties of Triangular Meshes," *IEEE Trans. Vis. Comput. Graphics*, vol. 18, no. 6, pp. 914–924, 2012.

[174] A. B. Yutaka and E. B. Y. Ohtake, "A Comparison of Mesh Smoothing Methods," in *In Proceedings of the Israel-Korea BiNational Conference on Geometric Modeling and Computer Graphics*, 2003, pp. 83–87.

[175] E. d'Eon, B. Harrison, T. Myers, and P. A. Chou, "8i Voxelized Full Bodies, version 2 – A Voxelized Point Cloud Dataset," ISO/IEC JTC1/SC29 Joint WG11/WG1 (MPEG/JPEG), Geneva, input document m40059/M74006, Jan. 2017.

[176] M. Krivokuća, P. A. Chou, and P. Savill, "8i Voxelized Surface Light Field (8iVSLF) Dataset," ISO/IEC JTC1/SC29 Joint WG11/WG1 (MPEG/JPEG), Ljubljana, input document m42914, Jul. 2018.

[177] Y. Xu, Y. Lu, and Z. Wen, "Owlii Dynamic Human Textured Mesh Sequence Dataset," ISO/IEC MPEG JTC1/SC29/WG11, Macau, China, Tech. Rep. m41658, Oct. 2017.

[178] C. Loop, Q. Cai, S. O. Escolano, and P. A. Chou, "Microsoft voxelized upper bodies - a voxelized point cloud dataset," ISO/IEC JTC1/SC29 Joint WG11/WG1 (MPEG/JPEG), input document m38673/M72012, May 2016.

[179] S. Lasserre and D. Flynn, "Improved TM3 for lossless and lossy compression of category 2 content," MPEG, Tech. Rep. m42519, 2018.

[180] D. R. Freitas, E. Peixoto, R. L. de Queiroz, and E. Medeiros, "Lossy Point Cloud Geometry Compression Via Dyadic Decomposition," in *2020 IEEE International Conference on Image Processing (ICIP)*, 2020, pp. 2731–2735.

[181] D. Flynn and K. Mammou, "G-PCC EE13.46 review of v11 attribute coding," MPEG, Tech. Rep. m55485, 2020.

[182] M. Quach, G. Valenzise, and F. Dufaux, "Improved Deep Point Cloud Geometry Compression," *2020 IEEE 22nd International Workshop on Multimedia Signal Processing (MMSP)*, pp. 1–6, 2020.

[183] J. Wang, D. Ding, Z. Li, and Z. Ma, "Multiscale Point Cloud Geometry Compression," *2021 Data Compression Conference (DCC)*, pp. 73–82, 2021.

[184] A. Guarda, N. M. M. Rodrigues, and F. Pereira, "Adaptive Deep Learning-based Point Cloud Geometry Coding," *IEEE Journal on Selected Topics in Signal Processing*, vol. 15, no. 2, pp. 415–430, Feb. 2021.

[185] A. Zaghetto, "Performance Analysis of Currently AI-based Available Solutions for PCC," ISO/IEC JTC 1/SC 29/WG 7, Tech. Rep. N0174, 2021.

[186] M. Quach, G. Valenzise, and F. Dufaux, "Learning convolutional transforms for lossy point cloud geometry compression," in *2019 IEEE international conference on image processing (ICIP)*. IEEE, 2019, pp. 4320–4324.

[187] J. M. Lounsbery, "Multiresolution Analysis for Surfaces of Arbitrary Topological Type," Phd Thesis, Dept. of Computer Science and Engineering, UW, Seattle, WA, USA, 1994.

[188] A. Khodakovsky, P. Schröder, and W. Sweldens, "Progressive geometry compression," in *SIGGRAPH*, USA, 2000, pp. 271–278.

[189] A. Khodakovsky and I. Guskov, "Normal mesh compression," *Geometric modeling for scientific visualization*, pp. 189–206, 2002.

[190] N. Dyn, D. Levine, and J. A. Gregory, "A butterfly subdivision scheme for surface interpolation with tension control," *ACM Trans. Graph.*, pp. 160–169, Apr. 1990.

[191] J.-Y. Sim, C.-S. Kim, C.-C. J. Kuo, and S.-U. Lee, "Rate-distortion optimized compression and view-dependent transmission of 3-D normal meshes," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 15, no. 7, pp. 854–868, 2005.

[192] F. Moran and N. Garcia, "Hierarchical coding of 3d models with subdivision surfaces," in *ICIP*, vol. 2. IEEE, 2000, pp. 911–914.

[193] W. Guan, J. Cai, J. Zhang, and J. Zheng, "Progressive coding and illumination and view dependent transmission of 3-D meshes using RD optimization," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 20, no. 4, pp. 575–586, 2010.

[194] MPEG WG7 3DGH, "Common Test Conditions for V-DMC," ISO/IEC JTC1/SC29/WG7, Geneva, Tech. Rep. w23081/N00685, Jul. 2023.

[195] ——, "V-DMC codec description - v4.0," ISO/IEC JTC1/SC29/WG7, Antalya, Tech. Rep. w22730/N00570, Apr. 2023.

[196] ——, "V-DMC TMM4.0," ISO/IEC JTC1/SC29/WG7, Antalya, Tech. Rep. w22759/N00595, Apr. 2023.

[197] I. Viola and P. Cesar, "Volumetric video streaming," in *Immersive Video Technologies*, G. Valenzise, M. Alain, E. Zerman, and C. Ozcinar, Eds. Elsevier, 2023, ch. 15, pp. 425–443.

[198] J. v. d. Hooft, M. T. Vega, T. Wauters, C. Timmerer, A. C. Begen, F. D. Turck, and R. Schatz, "From Capturing to Rendering: Volumetric Media Delivery with Six Degrees of Freedom," *IEEE Communications Magazine*, vol. 58, no. 10, pp. 49–55, Oct. 2020.

[199] Z. Liu, Q. Li, X. Chen, C. Wu, S. Ishihara, J. Li, and Y. Ji, "Point Cloud Video Streaming: Challenges and Solutions," *IEEE Network*, vol. 35, no. 5, pp. 202–209, Sep. 2021.

[200] M. Hosseini, "Adaptive Rate Allocation for View-Aware Point-Cloud Streaming," *arXiv*, Nov. 2017.

[201] M. Hosseini and C. Timmerer, "Dynamic Adaptive Point Cloud Streaming," in *Proceedings of the 23rd Packet Video Workshop*, ser. MMSys '18. ACM, Jun. 2018.

[202] J. Park, P. A. Chou, and J.-N. Hwang, "Volumetric Media Streaming for Augmented Reality," in *2018 IEEE Global Communications Conference (GLOBECOM)*. IEEE, Dec. 2018, pp. 1–6.

[203] ——, "Rate-Utility Optimized Streaming of Volumetric Media for Augmented Reality," *IEEE Journal on Emerging and Selected Topics in Circuits and Systems*, vol. 9, pp. 149–162, 2019.

[204] S. Petrangeli, G. Simon, H. Wang, and V. Swaminathan, "Dynamic Adaptive Streaming for Augmented Reality Applications," in *2019 IEEE International Symposium on Multimedia (ISM)*. IEEE, Dec. 2019, pp. 56–567.

[205] J. v. d. Hooft, T. Wauters, F. De Turck, C. Timmerer, and H. Hellwagner, "Towards 6DoF HTTP Adaptive Streaming Through Point Cloud Compression," in *Proceedings of the 27th ACM International Conference on Multimedia*, ser. MM '19. ACM, Oct. 2019, pp. 2405–2413.

[206] G. Cernigliaro, M. Martos, M. Montagud, A. Ansari, and S. Fernandez, "PC-MCU: point cloud multipoint control unit for multi-user holoconferencing systems," in *Proceedings of the 30th ACM Workshop on Network and Operating Systems Support for Digital Audio and Video*, ser. MMSys '20. ACM, Jun. 2020.

[207] J. Jansen, S. Subramanyam, R. Bouqueau, G. Cernigliaro, M. M. Cabré, F. Pérez, and P. Cesar, "A pipeline for multiparty volumetric video conferencing: transmission of point clouds over low latency DASH," in *Proceedings of the 11th ACM Multimedia Systems Conference*, ser. MMSys '20. ACM, May 2020.

[208] Z. Liu, J. Li, X. Chen, C. Wu, S. Ishihara, Y. Ji, and J. Li, "Fuzzy Logic-Based Adaptive Point Cloud Video Streaming," *IEEE Open Journal of the Computer Society*, vol. 1, pp. 121–130, 2020.

[209] S. Subramanyam, I. Viola, A. Hanjalic, and P. Cesar, "User Centered Adaptive Streaming of Dynamic Point Clouds with Low Complexity Tiling," in *Proceedings of the 28th ACM International Conference on Multimedia*, ser. MM '20. ACM, Oct. 2020, pp. 3669–3677.

[210] Y. Alkhalili, T. Gruczyk, T. Meuser, A. F. Anta, A. Khalil, and A. Mauthe, "Content-Aware Adaptive Point Cloud Delivery," in *2022 IEEE Eighth International Conference on Multimedia Big Data (BigMM)*. IEEE, Dec. 2022, pp. 13–20.

[211] M. Rudolph and A. Rizk, "View-Adaptive Streaming of Point Cloud Scenes through combined Decomposition and Video-based Coding," in *Proceedings of the 1st International Workshop on Advances in Point Cloud Compression, Processing and Analysis*, ser. MM '22. ACM, Oct. 2022, pp. 41–49.

[212] L. Wang, C. Li, W. Dai, S. Li, J. Zou, and H. Xiong, "QoE-Driven Adaptive Streaming for Point Clouds," *IEEE Transactions on Multimedia*, vol. 25, pp. 2543–2558, 2023.

[213] S. Gül, D. Podborski, T. Buchholz, T. Schierl, and C. Hellge, "Low-latency cloud-based volumetric video streaming using head motion prediction," in *Proceedings of the 30th ACM Workshop on Network and Operating Systems Support for Digital Audio and Video*, ser. NOSSDAV '20. New York, NY, USA: Association for Computing Machinery, Jun. 2020, pp. 27–33.

[214] S. Gül, D. Podborski, J. Son, G. S. Bhullar, T. Buchholz, T. Schierl, and C. Hellge, "Cloud rendering-based volumetric video streaming system for mixed reality services," in *Proceedings of the 11th ACM Multimedia Systems Conference*, ser. MMSys '20. New York, NY, USA: Association for Computing Machinery, May 2020, pp. 357–360.

[215] J. Son, Y. Sanchez, C. Hellge, and T. Schierl, "Split Rendering with L4S Over 5G for Latency Critical Interactive XR Applications," *IEEE Communications Magazine*, vol. 62, no. 8, pp. 46–52, Aug. 2024.

[216] T. Scandarolli, R. L. de Queiroz, and D. A. Florencio, "Attention-Weighted Rate Allocation in Free-Viewpoint Television," *IEEE Signal Processing Letters*, vol. 20, pp. 359–362, 2013.

[217] C. Dorea and R. L. de Queiroz, "Attention-Weighted Texture and Depth Bit-Allocation in General-Geometry Free-Viewpoint Television," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 27, no. 5, pp. 1055–1065, May 2017.

[218] MPEG WG7 3DGH, "Common Test Conditions for G-PCC," ISO/IEC JTC1/SC29/WG7, Sapporo, JP, output doc w24178/N944, Jul. 2024.

[219] ——, "Common Test Conditions for V-DMC," ISO/IEC JTC1/SC29/WG7, Sapporo, JP, output doc w24200/N964, Jul. 2024.

[220] H. Park, N. Kwon, and J. Byeon, "[V-DMC][EE4.10][SW] Integration of texture spatial scalability on TMMv9.0," ISO/IEC JTC 1/SC 29/WG 7, Kemer, input doc. m69845, Nov. 2024.

APPENDIX A

PUBLICATIONS

The research conducted in this thesis has led to some peer-reviewed publications, along with contributions to standardization efforts. Below is list a of all publications from the author:

Journal Papers

(i) M. Krivokuća,T. M. Borges, and R. L. de Queiroz, **Zerotree Coding of Subdivision Wavelet Coefficients in Dynamic Time-Varying Meshes,** in IEEE Transactions on Image Processing, vol. 34, pp. 1810–1819, 2025.

- Work developed during internship in InterDigital (Rennes, France), presented on Chapter 4.

(ii) T. M. Borges, R. U. B. Ferreira, D. C. Garcia, and R. L. de Queiroz, **Using Fractional Super-Resolution to Improve Lossy Compression of Point Cloud Geometry,** in SBrT/IEEE ComSoc Journal of Communication and Information Systems, vol. 38, no. 1, pp. 169–173, Nov. 2023.

- Practical application of the fractional super-resolution method to improve the quality of G-PCC compressed point clouds, covered in Chapter 3.

(iii) D. C. Garcia, A. Souto, G. Sandri, T. Borges, and R. Queiroz, **Point Cloud Reconstruction From Truncated Geometry-Based Streams,** in SBrT/IEEE ComSoc Journal of Communication and Information Systems, vol. 38, no. 1, pp. 77–84, Jul. 2023.

- Work developed with colleagues from the DIVP group, presenting a solution to add scalability to attributes using RAHT. Not covered in this thesis.

(iv) T. M. Borges, D. C. Garcia and R. L. de Queiroz, **Fractional Super-Resolution of Voxelized Point Clouds,** in IEEE Transactions on Image Processing, vol. 31, pp. 1380–1390, 2022.

- Work started during the author's Master's program, presented in Chapter 3.

(v) E. Alexiou, I. Viola, T. M. Borges, T. A. Fonseca, R. L. de Queiroz, and T. Ebrahimi, **A comprehensive study of the rate-distortion performance in MPEG point cloud compression,** in APSIPA Transactions on Signal and Information Processing, vol. 8, p. e27, 2019.

- Work developed during the author's Master's program, focusing on quality assessment of MPEG PCC. Selected for the 5th APSIPA Sadaoki Furui Paper Award.

Preprints

(i) T. M. Borges, Y. Sanchez, C. Hellge, and R. L. de Queiroz, **Optimizing Bit-Allocation of Multiple 3D Assets for Immersive-Viewpoint Applications,** paper currently under review.

- Work developed with colleagues from Fraunhofer HHI (Berlin, Germany), presented on Chapter 5.

(ii) T. M. Borges, T. E. de Campos, R. L. de Queiroz, **Towards robustness under occlusion for face recognition,** arXiv: 2109.09083 Sep. 2021.

- Study developed during the Computer Vision course, about applying GANs to enhance face recognition. Not covered in this thesis.

Contributions to Standardizations

➛ JVET

- S. Lee, S. Sasse, Y. Sanchez, R. Skupin, T. M. Borges, C. Hellge, T. Schierl (Fraunhofer HHI), **AHG9: Support for implicit representations with the Gaussian splatting information SEI message,** JVET-AN0264, Geneva, Sep. 2025.
- Y. Sanchez, R. Skupin, T. M. Borges, K. Sühring, C. Hellge, T. Schierl (Fraunhofer HHI), **AHG9: On general SEI payload constraints,** JVET-AM0174, Daejeon, Jun. 2025.
- R. Skupin, Y. Sanchez, A. Wieckowski, T. M. Borges, C. Hellge, T. Schierl (Fraunhofer HHI), **AHG9: Resolution nesting for FGC SEI message,** JVET-AL0211, Online, Mar. 2025.
- Y. Sanchez, T. M. Borges, R. Skupin, C. Hellge, T. Schierl (Fraunhofer HHI), **AHG9: On SEI processing order SEI message for HEVC and AVC,** JVET-AL0210, Online, Mar. 2025.
- T. M. Borges, Y. Sanchez, R. Skupin, C. Hellge, T. Schierl (Fraunhofer HHI), **AHG9: On providing robustness against layer dropping in multi-layer NNPF,** JVET-AL0209, Online, Mar. 2025.
- T. M. Borges, Y. Sanchez, R. Skupin, C. Hellge, T. Schierl (Fraunhofer HHI), **AHG9: On SPO root-process signaling constraint,** JVET-AL0208, Online, Mar. 2025.
- M. M. Hannuksela (Nokia), Y. Sanchez, T. M. Borges (Fraunhofer HHI), Y. Gao, P. Wu (ZTE), **AHG9: On SPO sub-chain signaling,** JVET-AK0333, Geneva, Jan. 2025.
- Y. Sanchez, T. M. Borges, R. Skupin, C. Hellge, T. Schierl (Fraunhofer HHI), **AHG9: On SPTI SEI message: Robustness and source constant framerate,** JVET-AK0168, Geneva, Jan. 2025.
- T. M. Borges, Y. Sanchez, R. Skupin, C. Hellge, T. Schierl (Fraunhofer HHI), **AHG9: On Enabling the usage of auxiliary layers for NNPF,** JVET-AK0167, Geneva, Jan. 2025.

- T. M. Borges, Y. Sanchez, R. Skupin, C. Hellge, T. Schierl (Fraunhofer HHI), **AHG9: On SPO sub-chain signaling,** JVET-AK0165, Geneva, Jan. 2025.
- T. M. Borges, Y. Sanchez, R. Skupin, C. Hellge, T. Schierl (Fraunhofer HHI), **AHG9: On SPTI SEI message,** JVET-AJ0252, Kemer, Oct. 2024.

➛ WG7 – MPEG 3DGH

- W. Zou, Y. Xu, H. Huang, V. Zakharchenko, H. Wei, S. Kuma, K. Hayashi, P. R. Alface, T. M. Borges, **[V-DMC][EE4.9] Report for EE 4.9 on base mesh generation for inter-frame,** ISO/IEC JTC 1/SC29/WG7, Sapporo, input doc. 68680, Jul. 2024.
- T. M. Borges, **[V-DMC][EE4.9] HHI Cross-check report for Test 3,** ISO/IEC JTC 1/SC29/WG7, Rennes, input doc. m67835, Apr. 2024.
- T. M. Borges, **[V-DMC][EE4.15] HHI Cross-check report for Test 1,** ISO/IEC JTC 1/SC29/WG7, Rennes, input doc. m67698, Apr. 2024.
- W. Zou, Y. Xu, H. Huang, V. Zakharchenko, H. Wei, S. Kuma, K. Hayashi, P. R. Alface, T. M. Borges, **[V-DMC][EE4.9] Report for EE 4.9 on base mesh generation for inter-frame,** ISO/IEC JTC 1/SC29/WG7, Rennes, input doc. m67449, Apr. 2024.
- T. M. Borges, **[V-DMC][EE4.19] HHI Cross-check report on Optimization for the geometry of the subdivided mesh,** ISO/IEC JTC 1/SC29/WG7, Rennes, input doc. m67416, Apr. 2024.
- T. M. Borges, **[V-DMC][EE4.15] HHI Cross-check report for Test 2,** ISO/IEC JTC 1/SC29/WG7, Rennes, input doc. m67339, Apr. 2024.
- M. Krivokuća, T. M. Borges, **[V-DMC] Clarifications on Zerotree Displacement Coding,** ISO/IEC JTC 1/SC29/WG7, Hannover, input doc. m65494, Oct. 2023.
- M. Krivokuća, T. M. Borges, **[V-DMC][EE4.7] Updated Zerotree Results for Wavelet Coefficients Coding,** ISO/IEC JTC 1/SC29/WG7, Geneva, input doc. m63677, Jun. 2023.
- M. Krivokuća, T. M. Borges, O. Roupin, **[V-DMC][EE4.7-related] Zerotree Coding Syntax for m63183,** ISO/IEC JTC 1/SC29/WG7, Antalya, input doc. m63186, Apr. 2023.
- M. Krivokuća, T. M. Borges, O. Roupin, G. Sandri, **[V-DMC][EE4.7-related] Hierarchical Zerotree Coding of Subdivision Wavelet Coefficients,** ISO/IEC JTC 1/SC29/WG7, Antalya, input doc. m63183, Apr. 2023.
- R. U. B. Ferreira, T. M. Borges, D. C. Garcia, and R. L. de Queiroz, **[G-PCC][EE13.56.a] Report on GPCC post-processing geometry super-resolution,** ISO/IEC JTC 1/SC29/WG7, Mainz, input doc. m60986, Oct. 2022.

- T. M. Borges, R. U. B. Ferreira, D. C. Garcia, and R. L. de Queiroz, **[G-PCC][EE13.56] Crosscheck report on Point Cloud Geometry Compression with a Hierarchical Prior,** ISO/IEC JTC 1/SC29/WG7, Online, input doc. m59361, Apr. 2022.
- R. U. B. Ferreira, T. M. Borges, D. C. Garcia, and R. L. de Queiroz, **[G-PCC] Post-processing geometry super-resolution based on lookup tables,** ISO/IEC JTC 1/SC29/WG7, Online, input doc. m58216, Oct. 2021.