ΟΙΚΟΝΟΜΙΚΟ ΠΑΝΕΠΙΣΤΗΜΙΟ ΑΘΗΝΩΝ
ATHENS UNIVERSITY OF ECONOMICS AND BUSINESS

School of Information Sciences and Technology
Department of Informatics
Athens, Greece

Master Thesis
in
Computer Science

Architectural Design Space for Real-Time Volumetric Streaming

Georgios Vidakis

Supervisor: Prof. George Xylomenos
Department of Informatics
Athens University of Economics and Business

Committee: Prof. Georgios Papaioannou
Department of Informatics
Athens University of Economics and Business

Prof. Vasilios A. Siris
Department of Informatics
Athens University of Economics and Business

January 2026

Georgios Vidakis

Architectural Design Space for Real-Time Volumetric Streaming

January 2026

Supervisor: Prof. George Xylomenos

Athens University of Economics and Business

School of Information Sciences and Technology

Department of Informatics

Mobile Multimedia Laboratory

Athens, Greece

Abstract

Volumetric video streaming enables interactive six-degrees-of-freedom experiences by delivering dynamic three-dimensional scene representations to remote users. Unlike conventional video streaming, volumetric media combines explicit geometric representations, high data volumes, intensive processing requirements, and strict interaction constraints. These characteristics introduce significant end-to-end latency challenges that directly impact responsiveness and immersion.

This thesis investigates latency reduction in real-time volumetric video streaming from a system-level architectural perspective. Rather than focusing on isolated compression or protocol optimizations, the study analyzes how architectural design decisions across the streaming pipeline influence cumulative latency behavior. After establishing the technical foundations of volumetric capture, representation, compression, and delivery, the thesis formulates an architectural taxonomy focused on latency-aware design, comprising client-adaptive delivery, cloud- and edge-rendered processing, native client-side decoding, multi-user shared delivery, and transport-level latency control mechanisms.

A comparative analysis of these architectural categories reveals distinct trade-offs between latency, visual quality, scalability, computational complexity, and deployment flexibility. The results show that no single architectural paradigm is sufficient to address all latency sources. Instead, effective volumetric streaming systems rely on hybrid and modular designs that integrate multiple latency mitigation strategies across representation, processing, and delivery layers.

By synthesizing existing approaches into a unified design space, this thesis provides structured insights into the architectural principles governing real-time volumetric streaming under strict system constraints. The findings contribute a systematic framework for understanding trade-offs and guiding the development of scalable, responsive, and practical real-time volumetric streaming systems.

Περίληψη

Η ογκομετρική μετάδοση βίντεο (volumetric video streaming) επιτρέπει την παροχή διαδραστικών εμπειριών έξι βαθμών ελευθερίας (6DoF), μέσω της απομακρυσμένης μετάδοσης δυναμικών τρισδιάστατων αναπαραστάσεων σκηνών. Σε αντίθεση με το συμβατικό δισδιάστατο ή πανοραμικό βίντεο, το ογκομετρικό περιεχόμενο βασίζεται σε ρητές γεωμετρικές αναπαραστάσεις, χαρακτηρίζεται από ιδιαίτερα μεγάλους όγκους δεδομένων, αυξημένες υπολογιστικές απαιτήσεις και αυστηρούς περιορισμούς διαδραστικότητας. Τα χαρακτηριστικά αυτά καθιστούν την καθυστέρηση (latency) κρίσιμο παράγοντα που επηρεάζει άμεσα την απόκριση και τον βαθμό εμβύθισης του χρήστη.

Η παρούσα διπλωματική εργασία εξετάζει τη μείωση της καθυστέρησης σε συστήματα ογκομετρικής μετάδοσης βίντεο πραγματικού χρόνου από αρχιτεκτονική και συστημική σκοπιά. Αντί να επικεντρώνεται σε μεμονωμένες τεχνικές συμπίεσης ή πρωτόκολλα μεταφοράς, η ανάλυση διερευνά πώς οι αρχιτεκτονικές επιλογές σε επίπεδο συστήματος επηρεάζουν τη συνολική, σωρευτική καθυστέρηση σε όλη την ακολουθία διαδικασιών της μετάδοσης. Αρχικά παρουσιάζονται τα θεμελιώδη στοιχεία της ογκομετρικής λήψης, ανακατασκευής, αναπαράστασης, συμπίεσης και απόδοσης (rendering). Στη συνέχεια διαμορφώνεται μία αρχιτεκτονική ταξινόμηση με επίκεντρο την καθυστέρηση, η οποία περιλαμβάνει προσεγγίσεις προσαρμοστικής μετάδοσης στην πλευρά του πελάτη, απομακρυσμένης απόδοσης σε υποδομές cloud ή edge, εγγενών αρχιτεκτονικών αποκωδικοποίησης στον πελάτη, κοινής και πολυχρηστικής διανομής περιεχομένου, καθώς και μηχανισμούς ελέγχου καθυστέρησης σε επίπεδο μεταφοράς.

Η συγκριτική ανάλυση των παραπάνω κατηγοριών αναδεικνύει θεμελιώδεις συμβιβασμούς μεταξύ καθυστέρησης, ποιότητας απεικόνισης, κλιμακωσιμότητας, υπολογιστικής πολυπλοκότητας και ευελιξίας ανάπτυξης. Τα αποτελέσματα καταδεικνύουν ότι καμία μεμονωμένη αρχιτεκτονική προσέγγιση δεν επαρκεί για την αντιμετώπιση όλων των πηγών καθυστέρησης. Αντιθέτως, τα αποδοτικά συστήματα ογκομετρικής μετάδοσης βασίζονται σε υβριδικούς και αρθρωτούς σχεδιασμούς που συνδυάζουν πολλαπλές στρατηγικές μείωσης της καθυστέρησης σε επίπεδο αναπαράστασης, επεξεργασίας και μεταφοράς.

Η εργασία αυτή συνθέτει τις υφιστάμενες προσεγγίσεις σε ένα ενιαίο αρχιτεκτονικό πλαίσιο και παρέχει δομημένες κατευθυντήριες αρχές για τον σχεδιασμό κλιμακούμενων και αποδοτικών

συστημάτων ογκομετρικής μετάδοσης πραγματικού χρόνου. Τα συμπεράσματα συμβάλλουν στην κατανόηση των αρχιτεκτονικών παραμέτρων που διέπουν τη διαχείριση της καθυστέρησης στο πλαίσιο ευρύτερων συστημικών περιορισμών και θέτουν τη βάση για μελλοντική έρευνα στον τομέα των διαδραστικών τρισδιάστατων μέσων.

Acknowledgements

I would like to express my sincere gratitude to my supervisor, Prof. George Xylomenos, for his guidance, insightful feedback, and continuous support throughout the development of this thesis. His expertise and academic perspective were invaluable in shaping both the direction and the depth of this work.

I would also like to thank the members of my thesis committee, Prof. Georgios Papaioannou and Prof. Vasilios Siris, for their time, constructive comments, and academic insight. Their work in the broader field of networking and distributed systems has been a source of inspiration during my studies.

I am grateful to the Department of Informatics at the Athens University of Economics and Business for providing a stimulating academic environment and the necessary resources to conduct this research.

Finally, I would like to thank my family and close friends for their constant encouragement and support throughout my studies in the Master's in Computer Science program. Their understanding and motivation played an essential role in the completion of this thesis.

Contents

1 Introduction

Volumetric video streaming aims to deliver dynamic three-dimensional scene representations that support interactive *six-degrees-of-freedom* (6DoF) experiences. Unlike conventional or view-based video formats, volumetric systems encode explicit scene geometry, enabling users to freely translate and rotate their viewpoint in a virtualized environment, thus requiring continuous viewpoint-dependent rendering.

This capability introduces a fundamental shift in system requirements. Volumetric content is characterized by significantly higher data rates, increased computational complexity, and strict responsiveness constraints driven by continuous user interaction. In contrast to traditional streaming systems, buffering cannot be used to mask delays without degrading the user experience, as any additional latency directly affects the motion-to-photon loop.

As a result, real-time volumetric streaming becomes a challenging system-level problem, where latency, data volume, computational cost, and scalability interact across multiple pipeline stages, including capture, reconstruction, compression, transmission, decoding, and rendering, as shown in Figure 1.1.

A key limitation in the existing literature is the lack of a clear distinction between fundamentally different application scenarios. In this thesis, two representative use cases are considered:

- **Immersive event streaming**, where volumetric content is captured using multi-camera setups and delivered to a large number of concurrent users. In this scenario, scalability and bandwidth dominate system design.
- **Real-time volumetric conferencing**, where a small number of participants capture and exchange volumetric data using consumer-grade RGB-D devices. Here, strict latency constraints and symmetric computational capabilities make both encoding and decoding critical bottlenecks.

These scenarios impose distinct system constraints and lead to different trade-offs. Treating them under a unified model can result in misleading conclusions regarding system performance and design.

This thesis adopts an architectural perspective to analyze the feasibility of real-time volumetric streaming systems under practical constraints. Rather than focusing on isolated

techniques, it examines how system-level design choices influence latency, data volume, and computational cost across the entire pipeline, and how different strategies redistribute these constraints.

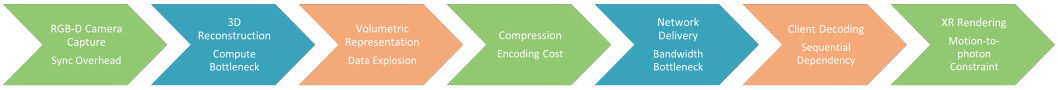


Fig. 1.1: High-level overview of the real-time volumetric streaming pipeline and its primary latency sources. The system focuses on RGB-D capture and interactive 6DoF consumption on resource-constrained XR devices, highlighting the key computational and communication bottlenecks across stages.

1.1 System Model

This thesis considers real-time volumetric streaming systems that support interactive 6DoF consumption under strict latency constraints. The system model captures the end-to-end pipeline from content generation to user rendering, focusing on the components that contribute to motion-to-photon latency.

At the capture stage, the system assumes RGB-D sensing, either through multi-camera setups or integrated depth-sensing devices. The captured data consists of synchronized color and depth streams, which are processed to reconstruct a time-varying three-dimensional representation of the scene. This reconstruction stage introduces significant computational overhead and may be executed either on dedicated servers or edge devices, depending on the application scenario.

The reconstructed content is represented using geometry-based formats such as point clouds or meshes and is subsequently compressed for transmission. The compression stage encodes both spatial structure (geometry) and associated attributes (e.g., color), producing a streamable representation of the scene.

The delivery stage transmits volumetric data over the network under strict latency constraints. Due to the interactive nature of the system, buffering is limited, and data must be delivered in a timely manner to support real-time rendering.

On the client side, the system assumes resource-constrained devices, such as XR headsets or mobile platforms. These devices are responsible for decoding the received volumetric data and rendering it according to the user's viewpoint. Rendering is viewpoint-dependent and must respond continuously to both translational and rotational motion, forming a tight interaction loop.

Latency in this system arises from multiple sources across the pipeline, including reconstruction cost, data volume, transmission delay, decoding complexity, and rendering

performance. These sources are interdependent, and delays introduced at earlier stages propagate through subsequent components.

The relative importance of each component depends on the application scenario. In conferencing systems, capture, reconstruction, and encoding are performed at the edge under strict latency constraints, often making encoding a dominant bottleneck. In contrast, event streaming systems can offload processing to centralized infrastructure, shifting the bottleneck toward data delivery, scalability, and client-side decoding.

This system model serves as the foundation for analyzing how design choices influence latency across different stages of the volumetric streaming pipeline.

1.2 Motivation and Problem Statement

Real-time volumetric streaming targets interactive scenarios where users explore a three-dimensional scene through continuous viewpoint changes. Unlike conventional video playback, the system must continuously adapt rendering to viewer motion, imposing strict constraints on motion-to-photon latency.

While the underlying volumetric content remains unchanged, rendering must be continuously updated based on the user's position and orientation, requiring low-latency access to sufficient scene information to support viewpoint-dependent reconstruction.

Existing research addresses these challenges by optimizing individual components of the pipeline. However, such approaches do not fully capture how design choices interact at the system level. Improvements in one stage may shift computational or bandwidth constraints to another, resulting in complex trade-offs. For example, scene compression reduces transmission latency, but increases encoding latency.

Furthermore, many existing approaches do not clearly distinguish between fundamentally different application scenarios, such as real-time conferencing and large-scale event streaming. As a result, they often optimize non-dominant bottlenecks or assume system constraints that do not hold in practical deployments.

This thesis addresses this gap by analyzing latency reduction from an architectural perspective. It examines how different approaches intervene at specific stages of the pipeline and how their combination affects overall system performance.

1.3 Objectives and Scope

The objective of this thesis is to provide a structured architectural analysis of real-time volumetric streaming systems, focusing on how different approaches enable practical deployment under strict system constraints. The analysis is grounded on the two representative application scenarios introduced earlier, which are used to contextualize architectural trade-offs and evaluate system feasibility.

Rather than optimizing a specific volumetric representation or codec, this work examines a range of system designs and identifies how different architectural approaches address latency across the streaming pipeline. These include data reduction techniques, computation offloading, representation and decoding optimizations, multi-user delivery strategies, and transport-level mechanisms.

The analysis aims to highlight the trade-offs between latency, visual quality, scalability, and computational complexity, and to derive design principles for building responsive volumetric streaming systems.

The scope of this thesis is limited to real-time applications. Offline playback, pre-rendered content, and non-interactive visualization scenarios are explicitly excluded, as they do not impose comparable latency constraints.

1.4 Research Questions

This thesis investigates latency reduction in volumetric video streaming from an architectural perspective. Rather than evaluating individual algorithms or codecs, the focus is placed on how system-level design choices influence latency behavior in real-world deployments. The research is guided by three main questions.

1. What architectural approaches are currently employed to enable real-time volumetric streaming under practical system constraints?

2. How do different architectural approaches trade latency against other critical system properties, such as visual quality, scalability, and computational complexity? This question addresses the inherent compromises that arise when designing interactive volumetric streaming systems.

3. What recurring architectural patterns and design principles can be extracted from existing systems? This question seeks to derive higher-level insights that can inform the design of future volumetric streaming platforms.

Together, these questions define the analytical scope of the thesis and provide a structured framework for evaluating and comparing volumetric streaming architectures.

1.5 Contributions

The main contribution of this thesis is a structured architectural analysis of real-time volumetric streaming systems, focusing on how different approaches address fundamental system constraints.

In addition, the thesis provides a breakdown of latency sources across the volumetric streaming pipeline and relates these sources to specific architectural choices. Through a comparative analysis of representative systems, the thesis highlights fundamental trade-offs between latency, visual quality, scalability, and computational complexity.

Finally, by synthesizing results from diverse volumetric streaming systems, the thesis derives a set of architectural insights that can guide the design of future real-time volumetric streaming platforms. These insights emphasize the importance of hybrid and multi-layer system designs in achieving consistent interactive performance. In particular, the thesis highlights that no single architectural approach is sufficient across all application scenarios, and that practical systems must balance latency, data volume, computation, and scalability through hybrid designs.

1.6 Thesis Structure

The remainder of this thesis is organized as follows. Chapter 2 introduces the fundamental concepts of volumetric video streaming, including capture, representation, compression, and rendering pipelines. Chapter 3 analyzes the architectural design space of latency reduction, classifying existing approaches based on their system-level intervention points. Chapter 4 provides a comparative analysis of these approaches under different application scenarios, highlighting trade-offs and design principles. Finally, Chapter 5 summarizes the main findings and outlines directions for future research.

2 Fundamentals of Volumetric Video Streaming

Volumetric video streaming refers to the real-time delivery of dynamic three-dimensional scene content for viewpoint-dependent rendering under six-degrees-of-freedom (6DoF) interaction. This chapter introduces the technical properties of volumetric data and the representations used in practice, emphasizing the aspects that directly affect compression, transmission, and real-time processing. These foundations are essential for understanding the challenges and trade-offs analyzed in later chapters.

2.1 From 2D and 360° Video to Volumetric Video

It is useful to clarify the conceptual difference between 2D video, 360° video, and volumetric video, since all three are often discussed under the broader category of immersive media delivery. In 2D video, the viewer consumes a sequence of frames from a single camera viewpoint. In 360° video, the viewer can freely rotate their viewpoint (yaw, pitch, roll), effectively looking around in all directions. However, the viewpoint remains fixed at the capture position, meaning that no new scene information is revealed through user motion. As a result, interaction is limited to rotational degrees of freedom (3DoF).

In contrast, volumetric video enables both rotational and translational movement along the X, Y, and Z axes, supporting full six-degrees-of-freedom (6DoF) interaction. This allows the user to change position within the scene, revealing previously occluded regions and introducing motion parallax. Supporting such interaction requires an explicit geometric representation of the scene, as the system must synthesize new viewpoints dynamically, rather than selecting from a predefined set of camera views.

A related concept is stereoscopic or multi-view video (often called 3D video), where two or more camera views provide depth cues through binocular disparity and limited motion parallax. However, these formats remain view-based rather than geometry-based: the viewpoint remains constrained to a predefined set of camera positions, without allowing free movement within the scene. Volumetric video differs in that it enables continuous viewpoint-dependent rendering based on an explicit geometric representation, rather than selecting among predefined views.

Tab. 2.1: Comparative overview of capture complexity, data volume, rendering cost, and degrees of freedom across 2D, 360°, and volumetric video formats.

Video	Cameras	Data Volume	Content Creation	Rendering	DoF
2D	Single-camera	Simple	–	Low	–
360°	Multi-camera	Medium	2D Stitch	Simple	3DoF
Volumetric	Single/Multi RGB-D	High	3D Reconstruction	Complex	6DoF

Table 2.1 summarizes the fundamental differences between conventional 2D video, 360° video, and volumetric video formats in terms of capture complexity, data volume, rendering overhead, and degrees of freedom. In practice, volumetric systems are commonly based either on multi-view camera setups or on one or more RGB-D sensing devices, which may internally use multiple sensors within a single integrated unit.

2.2 Volumetric Capture and Reconstruction Pipeline

Volumetric video systems follow a multi-stage pipeline that transforms raw sensor measurements into a streamable three-dimensional representation. This pipeline consists of capture, reconstruction, compression, transmission, and rendering stages, each introducing distinct computational and data-related challenges. Understanding the form of data at each stage is critical, as latency emerges from the interaction between these, components rather than from any single processing step.

Volumetric capture typically relies on multi-view or RGB-D sensing. In both cases, the output of the capture stage is not a volumetric representation, but a set of synchronized two-dimensional image streams. In RGB-D systems, each frame consists of multiple signals defined over a regular pixel grid, primarily a color image and a depth map, often accompanied by auxiliary measurements such as infrared or luminance data used internally for depth estimation. In practical systems, the color stream is commonly represented in a compressed-friendly format such as YUV with chroma subsampling, while the depth map is typically stored with higher precision (e.g., 16-bit per pixel) to preserve geometric accuracy. The color image encodes appearance, whereas the depth map provides a per-pixel distance measurement relative to the camera. In addition, camera calibration parameters (intrinsic and extrinsic) are required to relate pixel coordinates to the three-dimensional scene. Even so-called single volumetric cameras internally rely on multiple sensing elements (e.g., stereo pairs or structured-light modules), reinforcing that multi-view acquisition is fundamental to volumetric capture.

This representation remains fundamentally two-dimensional and structured. Each pixel corresponds to an independent measurement, and no explicit geometric connectivity or surface structure is provided at this stage. As a result, volumetric geometry must be reconstructed through a subsequent transformation process. Using camera calibration,

depth values can be mapped to three-dimensional coordinates, effectively converting each pixel into a 3D point associated with color attributes. This process transforms a structured image representation into an unstructured set of spatial samples. In multi-camera setups, multiple such streams are captured simultaneously from different viewpoints and must be temporally synchronized and geometrically aligned before reconstruction.

Volumetric reconstruction aggregates these per-view measurements into a unified three-dimensional representation. In RGB-D pipelines, this typically involves projecting depth pixels into 3D space and merging them across views using known camera poses. In multi-view systems without explicit depth sensing, reconstruction relies on correspondence estimation (e.g., feature matching and triangulation) to infer geometry. Regardless of the method, the reconstruction stage must resolve inconsistencies across views, remove redundant samples, and handle noise and missing data due to occlusions or sensor limitations.

A key property of current volumetric pipelines is that reconstruction is typically performed independently per frame, without enforcing temporal consistency. As a result, the generated 3D representation can vary significantly between consecutive frames, even for slowly changing scenes. This lack of stable correspondence across time increases data redundancy and limits the effectiveness of temporal compression techniques, in contrast to conventional video systems.

The output of the reconstruction stage is most commonly a point-based representation, where each frame is described as a set of discrete samples in three-dimensional space, optionally enriched with attributes such as color or normals. More structured representations, such as meshes, may be derived through additional processing, but maintaining consistent topology in dynamic scenes is challenging and computationally expensive. Consequently, many real-time systems favor point-based representations due to their flexibility and robustness to reconstruction artifacts.

To provide an order-of-magnitude estimate, consider a single RGB-D camera producing one color stream and one depth stream at 4K resolution (3840×2160). We assume 24-bit RGB color (8 bits per channel), a 16-bit depth map of the same spatial resolution, and no compression or auxiliary sensing streams, resulting in 40 bits per pixel. Each frame contains approximately 8.29 million pixels, corresponding to about 41.5 MB per frame.

At 30 frames per second, this results in a data rate of approximately 1.25 GB/s, or about 75 GB for one minute of uncompressed capture. At 60 frames per second, the data rate doubles to approximately 2.5 GB/s, corresponding to roughly 150 GB per minute.

These values show that even a single high-resolution RGB-D stream produces data volumes far beyond what can be transmitted directly in real time. Since they exclude multi-camera

overhead, auxiliary sensing streams, and metadata, they should be viewed as a simplified lower-bound estimate rather than a complete device-specific configuration.

Reconstruction is computationally intensive and constitutes a major contributor to end-to-end latency in live volumetric streaming systems [1]. The cost arises from both geometric processing (e.g., projection, alignment, fusion) and data management (e.g., handling large point sets). In practical deployments, reconstruction is often offloaded to dedicated servers or specialized hardware to meet real-time constraints. Nevertheless, its per-frame nature and computational complexity make it a fundamental bottleneck that directly impacts the feasibility of interactive volumetric applications.

2.3 Volumetric Representations

A volumetric streaming system must select a representation for its data. The representation can affect data volume, compression efficiency, decoding complexity, and rendering performance. Consequently, the choice of representation is not only a modeling decision, but a system-level design choice that determines whether real-time interaction is feasible.

Volumetric representations can be broadly categorized into explicit geometry-based approaches (e.g., point clouds and meshes), discrete volumetric grids (voxels), and more recent implicit or neural representations. Each category introduces different trade-offs between flexibility, computational cost, and suitability for dynamic real-time streaming.

2.3.1 Point Clouds

Point clouds represent a scene as a set of discrete samples in three-dimensional space, where each point is defined by spatial coordinates (x, y, z) and may include attributes such as color, normals, or confidence values. As the most direct form of 3D data, point clouds can be seen as the first explicit representation obtained after volumetric reconstruction. Most RGB-D cameras export some form of point cloud, for example, a rectangular buffer where every pixel has implicit (x, y) coordinates, a depth (z) and a color.

Their main advantage lies in the absence of explicit connectivity constraints. Since points are independent, there is no requirement to maintain consistent topology across frames, making point clouds particularly suitable for dynamic scenes with non-rigid motion, such as human capture [2].

However, this flexibility introduces important limitations. Point clouds lack an inherent surface representation, which can lead to visual artifacts such as holes or uneven density during rendering, especially when the viewpoint differs considerably from the viewpoint of the source camera. Furthermore, their irregular structure prevents the direct application

of grid-based compression techniques, making it difficult to exploit spatial and temporal redundancy.

From a system perspective, point clouds are often preferred in real-time volumetric streaming despite their high data volume, as they avoid the complexity and instability associated with topology reconstruction. This makes them a practical choice for latency-sensitive applications, where robustness is prioritized over compression efficiency.

2.3.2 Meshes

Meshes represent surfaces by connecting vertices into edges and faces, typically triangles. Compared to point clouds, they provide more visually coherent geometry and fit well with established graphics pipelines. However, in dynamic volumetric streaming, maintaining temporally stable mesh connectivity is difficult and computationally expensive. Reconstruction errors and motion can lead to frequent topology changes, complicating both compression and real-time processing. For this reason, although meshes can offer higher visual quality, many real-time systems favor point-based representations, which avoid explicit topology reconstruction. Also, since most RGB-D cameras output some form of point cloud, creating a mesh from this input requires considerable computational resources, to determine how the points (vertices) are connected to form surfaces.

2.3.3 Voxels and Hybrid Representations

Voxel-based representations discretize space into a regular three-dimensional grid. Their regular structure simplifies indexing and some processing operations, but the memory and bandwidth cost grows cubically with spatial resolution, making high-resolution dynamic content impractical for real-time streaming.

Hybrid representations reduce this cost by combining volumetric structure with more compact view-dependent or surface-based approximations. Such approaches trade full geometric generality for lower data volume, but they are not central to the systems analyzed in this thesis.

2.3.4 Emerging Representations

Recent approaches such as *Neural Radiance Fields* (NeRFs) and Gaussian Splatting represent scenes using continuous or semi-continuous models rather than explicit geometry. These methods can achieve high visual quality, but typically require substantial computation, memory, or scene-specific optimization, making them difficult to integrate into real-time interactive streaming systems with strict latency constraints. For this reason, the remainder

of this thesis focuses on explicit geometry-based representations, primarily point clouds, which remain the most practical option for latency-sensitive volumetric streaming.

2.4 Challenges of Volumetric Compression

The large data volume produced by volumetric capture pipelines makes compression essential for both storage and real-time transmission. However, volumetric data differs fundamentally from conventional image and video formats, limiting the effectiveness of existing compression techniques. In contrast to image-based media, volumetric representations are irregular, multi-dimensional, and often lack consistent spatial and temporal structure. These properties introduce significant challenges for efficient encoding, particularly in real-time streaming scenarios.

2.4.1 Why Conventional Video Compression Works

Conventional image compression methods, such as JPEG, operate on regular 2D image grids and exploit the strong spatial correlation between neighboring pixels. JPEG achieves compression by transforming image blocks into the frequency domain using the *Discrete Cosine Transform* (DCT), followed by quantization and entropy coding. This process is effective because natural images exhibit smooth variations and local redundancy, allowing the DCT to bring out the most important components of the image.

Video compression techniques, such as H.264 or HEVC, extend this principle by additionally exploiting temporal redundancy between consecutive frames, that is, predicting the blocks in each frame from blocks of the same size in other frames. Motion estimation (where the predicted block came from) and motion compensation (the difference between the predicted and actual block) allow encoding only frame differences, significantly improving compression efficiency.

2.4.2 Why It Fails for Volumetric Data

Regular video compression relies on two key assumptions, as explained above: (i) spatial locality on a regular grid, and (ii) stable temporal correspondence across frames. Volumetric data violates both assumptions.

First, spatial correlation is difficult to exploit because samples can be irregularly distributed in the three-dimensional space. In representations such as point clouds, neighboring elements in memory do not correspond to spatial neighbors, preventing the direct application of transform-based coding techniques such as DCT. But even when the point cloud is

essentially a rectangular frame with depth and color information, points at different depths may not exhibit the regularity of 2D images.

Second, temporal redundancy is significantly reduced. In dynamic volumetric content, geometry changes across frames and stable correspondence is often unavailable. Points may appear or disappear, and the reconstructed geometry may vary even for static regions, limiting the effectiveness of motion-based prediction.

Additionally, volumetric data exhibits sparsity and geometric discontinuities due to occlusions and sampling artifacts. These characteristics introduce abrupt variations that further reduce compressibility. As a result, conventional image and video compression techniques fail to efficiently exploit redundancy in volumetric data, motivating the need for specialized geometry-aware encoding methods.

2.4.3 Intraframe Coding Limitations

Intraframe coding refers to the compression of each frame independently, without exploiting temporal relationships between consecutive frames. In conventional video systems, intraframe coding is typically used for keyframes (I-frames), which serve as reference points for subsequent predictive frames (interframe coding).

Applying intraframe coding to volumetric data would require encoding each frame in full, including all geometric and attribute information. Given the large size of volumetric frames, this leads to extremely high data rates. Moreover, without temporal prediction, no information is reused across frames, resulting in poor compression efficiency.

As a result, intraframe-only approaches are generally insufficient for real-time volumetric streaming under strict bandwidth and latency constraints, motivating the development of specialized compression techniques that better exploit geometric structure and, where possible, temporal coherence.

2.5 Volumetric Compression Techniques

Given the limitations of conventional video compression, volumetric streaming systems rely on specialized techniques that explicitly account for irregular geometry and associated attributes. These techniques typically combine spatial data structures, predictive coding, and quantization strategies to reduce data size, while attempting to maintain real-time decoding performance.

2.5.1 Geometry and Attribute Compression

Volumetric compression typically separates geometry and attribute encoding due to their different statistical properties. Geometry compression focuses on efficiently representing spatial coordinates, while attribute compression targets associated data such as color or normals.

Geometry is often compressed using quantization and predictive coding. Coordinates are mapped to a discrete grid, reducing precision in exchange for lower storage cost. Predictive techniques then exploit local coherence by encoding differences between neighboring points instead of absolute positions. In structured representations, this prediction can significantly reduce entropy.

Attribute compression follows a similar principle but operates on irregularly distributed samples. In some cases, attributes are projected onto local neighborhoods or reordered to improve spatial correlation before applying transform-based coding or entropy encoding. However, unlike images, the lack of a regular sampling grid limits the effectiveness of these techniques.

This separation introduces additional overhead, as both geometry and attributes must be decoded and synchronized before rendering. Moreover, errors in geometry compression can propagate to attribute interpretation, further complicating efficient encoding.

2.5.2 Hierarchical Compression with Octrees

Octrees are a hierarchical data structure commonly used to represent and compress point cloud geometry. The core idea is to recursively subdivide 3D space into eight smaller regions (octants), starting from a bounding volume and continuing until a desired resolution is reached.

Each node in the octree represents a cubic region of space and stores whether that region is occupied or empty. By recursively subdividing only occupied regions, octrees provide an adaptive representation that avoids explicitly storing empty space, making them well suited for sparse volumetric data.

Compression is achieved by encoding the occupancy of each node, typically using binary flags, along with optional prediction schemes that exploit spatial coherence between neighboring nodes. This hierarchical representation enables significant data reduction, especially for large and sparse point clouds.

However, octree-based compression introduces several bottlenecks. Decoding requires traversing the tree structure from coarse to fine levels, creating strong sequential depen-

dencies. This limits parallelization and makes efficient GPU-based decoding difficult. On top of that, deep trees are required for high-resolution geometry, increasing both decoding latency and memory access overhead. As a result, while octrees are effective for compression, they often become a performance bottleneck in real-time volumetric streaming systems.

2.5.3 Practical Systems: Google Draco and MPEG PCC

Google Draco is a widely used open-source compression library designed for mesh and point cloud data [3]. It combines quantization, predictive coding, and entropy encoding to reduce both geometry and attribute size. For meshes, Draco exploits connectivity by predicting vertex positions based on neighboring vertices. For point clouds, it applies spatial reordering and prediction schemes to improve compressibility. In addition, although Draco can support relatively fast real-time pipelines compared to heavier codecs, encoding remains computationally expensive in symmetric live communication settings, where both endpoints must continuously compress outgoing geometry.

Despite its efficiency, Draco is primarily optimized for storage and offline transmission rather than real-time streaming. Its decoding pipeline involves multiple sequential steps, including bitstream parsing, prediction reconstruction, and entropy decoding, which are typically executed on the CPU. This limits parallelism and can lead to decoding bottlenecks, especially on mobile or XR devices.

MPEG *Point Cloud Compression* (PCC) represents a more standardized approach and is divided into two main categories: *Video-based PCC* (V-PCC) and *Geometry-based PCC* (G-PCC). V-PCC projects 3D point clouds onto multiple 2D planes and leverages existing video codecs for compression, effectively reusing mature video compression infrastructure. G-PCC, on the other hand, operates directly on point cloud data using hierarchical structures such as octrees.

While V-PCC can achieve high compression efficiency by leveraging video codecs, it introduces additional processing overhead due to projection and reconstruction steps, and may limit true 6DoF flexibility. G-PCC avoids projection but inherits the complexity and decoding limitations of hierarchical methods such as octrees.

Both approaches demonstrate that high compression efficiency often comes at the cost of increased computational complexity. In real-time volumetric streaming systems, this trade-off becomes critical, as encoding and decoding latency directly impact interactivity and user experience.

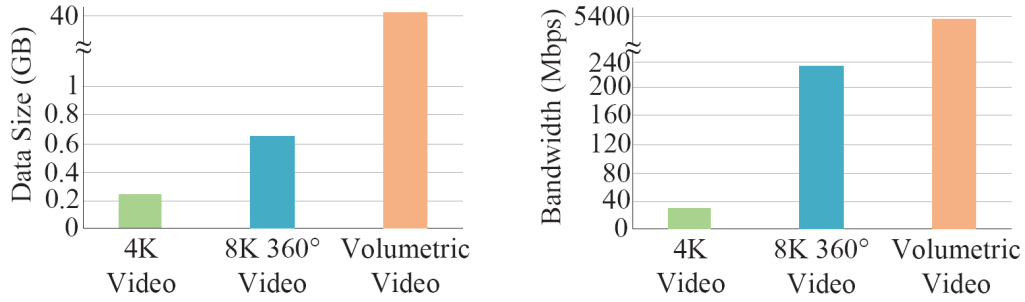
2.6 File Formats and Container Diversity

Beyond compression, volumetric streaming systems must define how geometry and attributes are packaged and delivered over time. Unlike conventional video, where standardized containers such as MP4 are widely used, volumetric media lacks a dominant format.

Existing formats, including point cloud (e.g., PLY, LAS) and mesh-based (e.g., OBJ, glTF), are primarily designed for storage and visualization rather than real-time streaming. As a result, they provide limited support for temporal sequencing, compression integration, and low-latency delivery.

Consequently, many systems rely on application-specific container formats that organize time-varying data into streamable segments, often including metadata such as timestamps or spatial partitioning information. While this enables flexibility in system design, it also increases complexity and limits interoperability across platforms.

This lack of standardization also complicates adaptive streaming, since segmentation, tile addressing, and metadata handling are often tightly coupled to application-specific implementations rather than common delivery abstractions.



(a) Data size for 1-minute Video **(b)** Required Bandwidth for Streaming

Fig. 2.1: Illustrative comparison of storage size and streaming bandwidth across conventional video formats and volumetric media, adapted from MetaStream [1]. The volumetric case is not tied to a single standardized resolution or codec, and depends strongly on scene representation, point density, and the specific compression pipeline used.

2.7 Data Volume and Bandwidth Requirements

One of the defining challenges of volumetric video streaming is the sheer volume of data generated by volumetric representations. Unlike conventional video, which encodes visual information on a 2D pixel grid, volumetric media must represent explicit three-dimensional geometry together with associated attributes such as color, normals, or other surface properties.

The resulting data size grows rapidly with scene complexity. For example, a single point cloud frame may contain hundreds of thousands to millions of points, each storing at least three spatial coordinates and additional attributes such as RGB color. Even with moderate precision (e.g., 16 bits per coordinate and 8 bits per color channel), the raw data size per frame can easily reach several megabytes.

When considering real-time streaming at typical frame rates (e.g., 30 frames per second), this translates to raw data rates on the order of hundreds of megabytes per second, far exceeding the bandwidth requirements of conventional video. While compression can significantly reduce this volume, the resulting bitrates remain substantially higher than those of traditional video streams.

The data rate is further influenced by several factors, including the chosen representation (e.g., point clouds vs. meshes), spatial sampling density, attribute richness, and compression efficiency. Higher fidelity representations increase both visual quality and bandwidth requirements, creating a direct trade-off between quality and deliverability.

Figure 2.1 illustrates this disparity by comparing typical bandwidth requirements across conventional video formats and volumetric media. A one-minute point cloud video sequence at 30 FPS can require over 60 times more data than an 8K 360° video and more than 200 times that of a 4K video at the same frame rate. In live streaming scenarios, the required bandwidth for raw point cloud video can be nearly 26 times higher than that of 8K 360° video and up to 216 times higher than 4K video.

These differences emphasize that volumetric streaming is not a simple extension of traditional video delivery, but a fundamentally more demanding problem in terms of data transmission and system design.

This mismatch between data generation and network capacity makes bandwidth a primary bottleneck in volumetric streaming systems. Unlike traditional streaming, where latency is often dominated by network variability, volumetric systems are fundamentally constrained by the volume of data that must be transmitted. Consequently, system design must focus on reducing transmitted data through compression, representation choices, and adaptive streaming strategies, rather than relying solely on network-level optimizations.

2.8 Summary

This chapter introduced the fundamental characteristics of volumetric video streaming, including data acquisition through multi-view or RGB-D capture, common volumetric representations, and the challenges associated with compressing and delivering such data in real time.

It examined how the structure of volumetric data differs from conventional image and video formats, highlighting the limitations of traditional compression techniques and the need for specialized approaches. In addition, the significantly higher data volume and bandwidth requirements of volumetric content were discussed as key constraints for practical system design.

These foundations motivate the architectural analysis of the next chapter, where real-time volumetric streaming is examined as a system-level feasibility problem shaped by interacting bottlenecks across the pipeline.

3 Architectural Design Space for Real-Time Volumetric Streaming

Enabling low latency real-time volumetric streaming cannot be achieved through a single optimization technique. Existing systems instead intervene at different points of the end-to-end pipeline, each targeting a specific bottleneck, such as data volume, encoding and decoding cost, network behavior, or multi-user scalability.

In interactive 6DoF systems, latency emerges from the combined effect of multiple system components, including processing, transmission, and rendering. While these challenges are also present in conventional streaming and conferencing systems, they are amplified in volumetric media due to higher data rates and stricter interaction requirements.

This chapter organizes existing volumetric streaming systems based on their Architectural Intervention Point, defined as the stage or mechanism through which latency is primarily reduced. Rather than classifying approaches by implementation details or specific algorithms, this perspective focuses on how each system restructures the flow of data and computation to improve responsiveness.

The analyzed systems represent distinct intervention strategies, including data reduction at the delivery stage, computation relocation to cloud or edge infrastructure, representation and decoding optimizations, scalability-aware multi-user delivery, and transport-level latency control. Each category targets a different source of delay and introduces specific trade-offs between latency, visual quality, computational complexity, and system scalability. This approach enables a systematic comparison of how different design choices contribute to the practical feasibility of real-time volumetric streaming under different application requirements.

3.1 Use Case Context: Event Streaming vs Conferencing

The two use cases considered in this work, immersive event streaming and real-time volumetric conferencing, were introduced in Section 1.1. While both involve real-time volumetric data delivery, they differ significantly in their dominant system constraints.

In immersive event streaming, the primary challenge lies in scalable data distribution to a large number of users, making bandwidth efficiency and multi-user delivery the central concerns. In contrast, volumetric conferencing involves a small number of participants operating under symmetric constraints, where end-to-end latency and computational efficiency, particularly during encoding, become the dominant factors.

This distinction is used throughout the following sections to contextualize architectural design choices and evaluate their applicability across different volumetric streaming scenarios.

3.2 Latency Sources in Volumetric Streaming Pipelines

Latency in volumetric streaming emerges from a set of distinct system-level bottlenecks rather than from the presence of multiple pipeline stages. In the multi-camera 6DoF setting considered in this thesis, these bottlenecks arise from the interaction between data generation, transmission, and processing constraints, each contributing to the overall *Motion-to-Photon* (M2P) delay.

Capture and Reconstruction Bottleneck. The primary latency contributor during content generation is the synchronization and fusion of multiple RGB-D streams. In multi-camera setups, frames must be temporally aligned before reconstruction; even minor asynchrony introduces processing stalls or inconsistencies in the synthesized geometry. Beyond synchronization, reconstruction requires merging partially overlapping views and removing redundant points across camera *Fields of View* (FoVs), a process that introduces significant computational overhead. Systems such as MetaStream demonstrate that point cloud fusion and redundancy elimination dominate early-stage latency, particularly in live scenarios where processing must be performed continuously.

Data Volume and Transmission Bottleneck As discussed in Section 2, volumetric representations generate data rates that exceed conventional video by orders of magnitude. This creates a throughput bottleneck in which delay is determined not only by network capacity, but also by the amount of geometry and attributes that must be delivered before useful rendering can occur. This bottleneck is fundamentally more severe than in conventional video streaming, as volumetric representations lack efficient inter-frame prediction and often require near-complete data reception for correct reconstruction, limiting the effectiveness of partial delivery. In interactive 6DoF systems, this bottleneck is exacerbated by the inability to rely on large playback buffers: buffering can mitigate network jitter but introduces a constant delay that directly increases M2P latency. Consequently, delivery

mechanisms must operate under tight latency constraints while handling extremely high data volumes.

Encoding Bottleneck In real-time volumetric systems, encoding can become a dominant source of latency, particularly in conferencing scenarios where both endpoints must continuously compress captured data. Point cloud compression algorithms involve geometry processing, attribute encoding, and spatial partitioning, all of which are computationally intensive. Unlike decoding, which can often be parallelized or offloaded, encoding must be performed online and under strict latency constraints, making it a critical bottleneck in symmetric streaming architectures.

Decoding Bottleneck A critical source of latency arises from the structure of volumetric compression formats. As established in Section 2, octree-based representations (e.g., Google Draco) exploit spatial hierarchy for compression efficiency but introduce strict serial dependencies during decoding. Specifically, the position of each node depends on its parent, forcing recursive traversal of the tree. This limits parallelization on GPUs and leads to underutilization of available hardware resources. On mobile and XR devices, this decoding model can significantly constrain frame rate and responsiveness, particularly on resource-constrained hardware.

Rendering and Interaction Bottleneck In client-rendered architectures, latency is further amplified by the requirement for a tight motion-to-photon loop. Viewpoint-dependent rendering must respond immediately to both translational and rotational user motion, leaving minimal tolerance for accumulated delay. Even when preceding stages are optimized, rendering latency can become dominant due to limited GPU resources and the need to maintain frame rates of at least 30 FPS on thermally constrained devices. In remote-rendering architectures, this bottleneck shifts away from local rendering and toward network latency and server-side processing.

These bottlenecks are interdependent and cumulative: delays introduced during reconstruction increase the amount of data to be processed downstream, while large data volumes amplify both transmission and decoding cost. As a result, latency cannot be attributed to a single component but must be understood as the outcome of interacting constraints across the system. This observation motivates the architectural intervention strategies analyzed in the following section, where each approach targets a specific bottleneck within the pipeline.

3.3 Architectural Intervention Point Taxonomy

Real-time volumetric streaming is enabled through targeted interventions at specific points of the system, rather than through a single global optimization. Each architectural approach

improves responsiveness by addressing a particular bottleneck identified in the above section, effectively "breaking" the propagation of delay within the pipeline. To capture this behavior, existing systems are classified according to their *Architectural Intervention Point*, defined as the component or mechanism where the system primarily intervenes to improve responsiveness or reduce a dominant bottleneck. This perspective emphasizes how latency is redistributed across the system.

Based on this principle, five main categories of intervention are identified:

- *Data Reduction (Client-Adaptive Delivery):* These approaches reduce latency by limiting the amount of volumetric data that must be transmitted and processed. By selecting only the subset of geometry that contributes to the final rendered view, they directly address the data volume and transmission bottleneck.

- *Computation Relocation (Cloud/Edge Rendering):* These architectures shift computationally intensive tasks away from the client device to remote infrastructure. By offloading decoding and rendering, they eliminate client-side processing bottlenecks, at the cost of increased dependence on network latency and interaction delay.

- *Representation and Decoding Optimization:* This category targets the decoding bottleneck by redesigning data structures and processing pipelines to remove serial dependencies and enable parallel execution. The goal is to improve hardware utilization, particularly on GPUs, and reduce decoding latency on resource-constrained devices.

- *Scalability-Aware Delivery (Multi-User Sharing):* These approaches address latency under high concurrency by exploiting redundancy across users. By sharing computation or transmitted data among multiple clients, they reduce per-user processing and delivery overhead, improving scalability while introducing trade-offs in personalization.

- *Transport-Level Control:* These mechanisms operate at the network layer to stabilize delivery behavior and minimize latency introduced by congestion and queueing. They target network-induced variability and aim to maintain a predictable and low-delay interaction loop.

These categories are not mutually exclusive. In practice, modern volumetric streaming systems combine multiple intervention strategies, integrating data reduction, computation offloading, decoding optimization, and transport control to address the diverse sources of latency identified earlier.

The following sections analyze each category in detail, examining how specific systems implement these interventions and the trade-offs they introduce.

3.4 Visibility-Aware and Saliency-Driven Data Reduction

Visibility-aware architectures mitigate the data volume bottleneck by selectively transmitting only the subset of volumetric content that contributes to the final rendered view. Instead of reducing latency through faster processing, these approaches limit the amount of data that must be delivered and decoded, effectively reducing both transmission delay and client-side workload.

3.4.1 Viewport-Adaptive Fetching (ViVo)

The ViVo system [4] addresses the fundamental challenge of volumetric video streaming: the prohibitive size of point cloud data. Instead of relying on more efficient compression, ViVo follows a different design principle: *not all data needs to be transmitted*. Specifically, the system reduces the transmitted data volume by delivering only the subset of the scene that contributes to the user's current view.

From a system perspective, ViVo operates at the delivery stage of the streaming pipeline, targeting the data volume bottleneck. Rather than modifying the encoding or decoding processes, it performs *data selection*, determining which parts of the volumetric content should be transmitted at each time step. To enable this, the volumetric scene is partitioned into spatial tiles. For each frame, the system predicts a set of tiles $\hat{L}$ that are likely to be required based on the user's viewpoint. Only these tiles are fetched and transmitted, while the rest are discarded.

Prediction accuracy is measured using the Jaccard Index between the predicted tile set $\hat{L}$ and the actual set of tiles consumed by the user L. This similarity is incorporated into an *Exponentially Weighted Moving Average* (EWMA), producing a stability-aware estimate S that reflects the reliability of the prediction over time. This feedback signal directly controls the aggressiveness of data reduction. When S is high, indicating stable and accurate predictions, the system aggressively prunes *Out-of-Sight* (OOS) tiles, achieving bandwidth savings of up to 80%. When prediction accuracy decreases, the system becomes more conservative by increasing redundancy, ensuring that sudden viewpoint changes do not result in missing geometry.

Tile selection is further refined through three complementary visibility mechanisms:

- Viewport Visibility: selects tiles that lie within the user's viewpoint.
- Occlusion Visibility: excludes tiles that are hidden by foreground geometry.

- Distance Visibility: prioritizes tiles based on their distance from the user, reflecting their relative visual contribution.

By combining prediction with geometric visibility filtering, ViVo transforms tile delivery into a closed-loop control process. Bandwidth allocation continuously adapts to both user behavior and scene structure, enabling efficient streaming without requiring changes to the underlying compression pipeline.

3.4.2 Hybrid Saliency-Based Tiling

While ViVo relies on geometric visibility and viewpoint prediction, hybrid saliency-based approaches [5] extend data reduction by incorporating perceptual importance into the streaming process. The key idea is that not all visible regions contribute equally to the user's experience; therefore, data transmission can be further reduced by prioritizing perceptually important content.

Visual saliency refers to the likelihood that a user will focus their attention on specific regions of a scene. In the context of volumetric video, saliency captures both structural importance (e.g., geometric complexity) and temporal relevance (e.g., motion or interaction).

From a system perspective, this approach operates at the delivery stage of the pipeline and targets the data volume bottleneck, similar to ViVo. However, instead of selecting data purely based on visibility, it performs *importance-aware data selection*, allocating resources according to perceptual relevance.

To estimate saliency, the system models two complementary components. Static saliency is computed using Simplified *Point Feature Histograms* (SPFH), which encode local geometric structure through angular relationships between surface normals. This allows the system to identify regions with high curvature or structural variation, which are more likely to attract user attention.

Dynamic saliency is derived from inter-frame variability, capturing regions with motion or temporal changes. These regions are typically more perceptually significant, as motion strongly influences visual attention. By combining static and dynamic cues, the system assigns a saliency score S_V to each spatial unit, reflecting both geometric distinctiveness and temporal importance.

The key architectural mechanism is a saliency-driven tiling strategy based on hierarchical clustering. Instead of using uniform spatial partitioning, fine-grained tiles are grouped into clusters according to saliency similarity. This results in a content-aware representation, where tile boundaries adapt to perceptually important regions. This design enables

differentiated data allocation: high-saliency clusters are transmitted at higher quality or priority, while low-saliency regions can be downsampled or omitted. As a result, the system achieves more efficient bandwidth utilization by aligning transmission decisions with human visual perception, rather than purely geometric visibility.

3.4.3 Discussion

Both approaches reduce latency by minimizing the volume of transmitted and decoded data, transforming delivery into a selective process based on visibility or perceptual importance. However, this efficiency depends on estimation mechanisms (e.g., viewpoint prediction or saliency modeling), introducing a trade-off between data reduction and robustness to uncertainty.

3.5 Cloud- and Edge-Rendered Volumetric Streaming Architectures

Cloud- and edge-rendered architectures address the high computational demands of volumetric video streaming by fundamentally restructuring the processing pipeline. Instead of transmitting volumetric representations (e.g., meshes or point clouds) for local decoding and rendering, these systems render the scene on remote infrastructure and stream only the resulting 2D view to the client [6, 7].

The core design principle is that *the client does not need access to the full 3D representation if it only consumes its visual projection.* By collapsing the volumetric pipeline into a 2D video delivery problem, these systems eliminate the need for geometry decoding and rendering on resource-constrained devices.

From a system perspective, this approach operates at the rendering stage of the pipeline and targets the computational bottleneck. Rather than reducing data size or improving compression, it performs *computation offloading*, relocating volumetric processing to GPU-equipped servers.

3.5.1 Server-Side Volumetric Rendering

In the system proposed by Gül et al. [6], volumetric content is stored as pre-compressed geometry and texture streams. On the server, these streams are decoded and reconstructed into a 3D scene representation. A rendering engine (e.g., Unity) generates a view-dependent image based on the user's current head pose.

The rendered frames are then encoded using hardware video encoders and transmitted via low-latency streaming protocols such as WebRTC. On the client side, the system reduces to a conventional video playback pipeline, where frames are decoded and displayed while head pose updates are continuously sent back to the server.

This architecture transforms volumetric streaming into a closed interaction loop in which the client transmits head pose updates, the server renders the corresponding viewpoint, and the resulting image is streamed back to the client.

Unlike data-driven approaches such as ViVo, where partial scene data is transmitted, this design completely removes volumetric data from the client-side pipeline, replacing it with a continuous stream of rendered images.

3.5.2 Latency-Aware Rendering through Head Motion Prediction

A fundamental challenge in remote rendering is that the rendered frame is always based on a past user pose due to network and processing delays. To address this, Gül et al. [7] introduce a prediction-driven rendering mechanism.

Instead of rendering the scene for the current pose, the system predicts a future pose $\hat{P}(t + \Delta)$, where Δ corresponds to the estimated motion-to-photon latency. This prediction is performed using autoregressive models or Kalman filtering over recent pose trajectories.

By rendering for the predicted pose, the system attempts to align the displayed frame with the user's actual viewpoint at display time. This effectively shifts the rendering process forward in time, transforming latency compensation into a prediction problem.

3.5.3 Image-Space Correction through Post-Rendering Warping

Since prediction errors are unavoidable, a second layer of correction can be applied on the client side through post-rendering warping [8]. This mechanism operates directly in the 2D image space and does not require access to the underlying 3D geometry.

Upon receiving a rendered frame, the client computes the difference between the predicted pose used at the server and the latest available head pose. This transformation is approximated using a planar homography, which defines a mapping between the two viewpoints.

The received image is then warped accordingly, producing a corrected frame that better matches the user's current perspective. To support this process, the server renders an overscanned image with a slightly larger field of view, ensuring that sufficient pixel information is available after the transformation.

This approach enables low-cost correction of small viewpoint discrepancies without requiring additional server-side rendering or increased bandwidth.

3.5.4 Discussion

These approaches transform volumetric streaming into a remote rendering pipeline, where volumetric processing is centralized and the client consumes only 2D video. Their effectiveness relies on prediction and image-space correction mechanisms that compensate for network and processing delay, rather than eliminating it.

3.6 Representation Design and Neural Enhancement

Representation-aware architectures address latency by modifying how volumetric content is represented, processed, and reconstructed across the pipeline. Unlike visibility-driven approaches (Section 3.4), which reduce the amount of transmitted data, these methods target the processing bottlenecks of volumetric streaming, either by enabling efficient decoding or by shifting reconstruction complexity across system components.

3.6.1 Parallel Decoding through Representation Design (GROOT)

The GROOT system [2] targets the client-side decoding bottleneck, which arises from the inherent structure of point cloud compression schemes such as octrees. In conventional octree-based representations, spatial information is encoded hierarchically, where each node depends on its parent. This creates strong *serial dependencies*, forcing recursive traversal during decoding and limiting parallel execution. GROOT addresses this limitation through a redesigned representation called the *Parallel Decodable Tree* (PD-Tree). The key idea is to preserve the compression efficiency of octrees while eliminating decoding dependencies in the most computationally intensive parts of the structure.

Specifically, the octree is partitioned at a predefined depth D_b into two components:

- Octree Breadth Bytes (OBB): The upper levels of the tree are encoded in breadth-first order and decoded sequentially on the CPU, providing a coarse representation of the scene.

- Octree Depth Bytes (ODB): The deeper levels are reorganized such that each leaf node is encoded independently as a path from depth D_b to the leaf. This removes inter-node dependencies and enables parallel decoding across GPU threads.

This hybrid design ensures that only a small portion of the structure remains sequential, while the majority of the decoding workload can be executed in parallel. As a result, geometry decoding can be offloaded to the GPU and integrated directly into the rendering pipeline (e.g., via vertex shaders), significantly reducing decoding latency.

In addition to geometry representation, GROOT identifies color encoding as a major bottleneck in existing point cloud codecs. After geometry compression, color data can dominate the overall bitrate due to poor spatial locality. To address this, GROOT reorganizes color information into a 2D layout and applies image-based compression (e.g., JPEG), using locality-preserving orderings (e.g., Morton order) to improve compression efficiency without increasing decoding complexity.

As a result, GROOT achieves stable real-time performance on mobile devices, reaching approximately 30 FPS, whereas traditional Draco-based pipelines are typically limited to 10–15 FPS due to sequential decoding constraints.

Overall, GROOT operates at the representation and decoding stages of the pipeline, reducing latency by enabling parallel processing rather than reducing transmitted data.

3.6.2 Neural Enhancement through 3D Super Resolution (VoluSR)

VoluSR [9] addresses the data volume bottleneck of volumetric streaming by shifting part of the reconstruction process to the client. Instead of transmitting high-density point clouds, the system delivers a low-resolution representation and reconstructs high-resolution geometry locally using 3D *Super-resolution* (SR). This design reduces bandwidth consumption significantly (up to 74%), at the cost of additional computation on the client device. However, applying existing SR models directly is not feasible in real-time settings. Models such as PU-GAN achieve only $\sim$0.1 FPS even on high-end GPUs, while also introducing temporal inconsistencies between consecutive frames and lacking native support for color. As a result, SR must be re-engineered as part of the overall streaming system rather than treated as an independent component.

At a high level, SR operates on local patches of the point cloud through three stages: feature extraction, feature expansion, and point generation. Profiling shows that feature extraction dominates the computational cost (approximately 78% of inference time), making it the primary bottleneck. VoluSR addresses this by simplifying the model and redesigning how local neighborhoods are processed. In particular, it replaces conventional neighborhood-based convolutions with spherical kernel functions, which organize neighboring points into structured angular and radial regions. This reduces the need for expensive nearest-neighbor queries and enables more efficient feature aggregation while preserving geometric structure.

Additional optimizations further reduce computational overhead. The model is trimmed by removing redundant layers and limiting feature dimensionality, while patch generation is simplified using structured spatial partitions instead of overlapping kNN-based regions. These changes significantly improve throughput and reduce memory usage, bringing SR closer to real-time performance.

To address temporal instability, VoluSR exploits the strong correlation between consecutive frames. Instead of performing full SR inference on every frame, the system estimates motion in the low-resolution domain and propagates previously reconstructed high-resolution results accordingly. This reduces redundant computation and improves temporal consistency by maintaining smoother point trajectories across frames.

VoluSR also leverages the unordered nature of point clouds by merging the low-resolution input with the SR output. This allows lower upsampling ratios to approximate higher-resolution results, effectively trading reconstruction accuracy for reduced computational cost. Finally, the system introduces a joint adaptation mechanism that selects both the transmitted data quality and the SR upsampling ratio at runtime. Increasing SR reduces bandwidth usage but increases computational load, while decreasing SR has the opposite effect. This trade-off is managed through a QoE-driven optimization that balances visual quality, latency, and resource constraints.

Overall, VoluSR transforms volumetric streaming into a reconstruction-aware pipeline, where part of the scene is transmitted explicitly and part is inferred at the client. By shifting complexity from the network to the device, it provides an alternative to purely compression-based solutions, enabling efficient streaming under bandwidth-constrained conditions.

3.6.3 Discussion

These approaches illustrate two complementary strategies: reducing processing cost through optimized representation design (GROOT), and reducing transmitted data through client-side reconstruction (VoluSR). In both cases, latency reduction is achieved by shifting the bottleneck across system resources, rather than eliminating it.

3.7 Scalability-Aware Multi-User Delivery

Scalability-aware architectures address a fundamental limitation of volumetric streaming systems: the coupling between resource consumption and the number of concurrent users. In conventional designs, each client requires a dedicated processing pipeline—whether through independent rendering, encoding, or data delivery—causing computation and bandwidth costs to scale linearly with user count.

To mitigate this, these approaches restructure the delivery pipeline to *share resources across users*. Instead of treating each viewer as an independent session, they exploit redundancy in either user viewpoints or transmitted data, enabling reuse at different stages of the system. This results in architectures where the dominant cost depends on shared representations or grouped users, rather than individual clients.

3.7.1 Transcoded View Sharing and Content Hybridization (MuV2)

The MuV2 system [10] addresses the scalability limitations of multi-user volumetric streaming systems that rely on server-side rendering and transcoding. In such architectures, volumetric content is decoded and rendered at the edge, and each user receives a view-dependent 2D stream. While this approach reduces client-side computation, it introduces a new bottleneck: the rendering and encoding cost scales linearly with the number of users, as each viewpoint requires a separate rendering and video encoding pipeline.

MuV2 tackles this problem at the rendering and encoding stages of the pipeline by reducing the number of unique views that must be generated and transmitted. Instead of treating each user independently, the system exploits redundancy across users observing the same scene, reformulating multi-user delivery as a resource allocation problem across computation and bandwidth.

A first component of the system is *content hybridization*, which balances the use of volumetric and view-dependent representations. Rather than uniformly streaming full 3D data or fully relying on server-side rendering, MuV2 dynamically assigns the representation type per user based on their spatial relationship to the content. Users located close to the object, where geometric fidelity is critical, receive volumetric data, while users positioned further away are served with edge-rendered 2D views. This hybrid strategy allows the system to distribute load between network bandwidth and edge computation, avoiding the saturation of either resource.

The core mechanism enabling scalability is *cross-user view sharing*. MuV2 clusters users with similar viewpoints and assigns a shared reference view to each cluster. This clustering is formulated as a K-median optimization problem, where the objective is to minimize the perceptual distortion introduced by approximating individual viewpoints with a common rendered view. The distortion is explicitly modeled as a function of viewpoint displacement and scene geometry, allowing the system to bound the quality degradation introduced by sharing.

For each cluster, the edge server renders a single RGB-D view, which is then transmitted to all users in the group. Each client reconstructs its exact viewpoint locally through image warping, using the depth information to approximate the correct geometry. This transforms the rendering cost from a per-user operation to a per-cluster operation, effectively reducing

complexity from $O(N)$ to $O(K)$, where K is the number of clusters and typically much smaller than the number of users.

To further address scalability at the encoding stage, MuV2 introduces an encoder multiplexing mechanism. Since hardware video encoders (e.g., NVENC) can only support a limited number of concurrent streams, multiple view streams are time-multiplexed on the same encoder. This is achieved by aligning *Group-of-Pictures* (GOP) boundaries across streams, enabling deterministic scheduling of encoding tasks without increasing end-to-end latency. As a result, the encoding workload becomes bounded by the number of available encoders rather than the number of users.

By combining hybrid representation selection, shared rendering, and multiplexed encoding, MuV2 reduces both network and computation costs in multi-user scenarios. The system demonstrates real-time performance with over 30 concurrent users at 30 FPS on a single edge server, showing that server-side rendering can be partially scaled through view sharing.

3.7.2 Encoding-Level Scalability through Multiple Description Coding (MDC)

The MDC-based architecture [11] addresses scalability at the encoding stage by eliminating the dependency between the number of users and the number of encoded streams. In conventional one-to-many volumetric streaming systems, each client requires a separate representation tailored to its bandwidth, leading to a preprocessing cost that grows linearly with the number of users. Even with parallel encoding, this quickly saturates available CPU resources and reduces the achievable frame rate.

MDC removes this dependency by restructuring each volumetric frame into a small set of independently encoded *descriptions*. Instead of encoding one representation per user, the system encodes a fixed number K of descriptions (typically $K = 3$), each corresponding to a different sampled subset of the original point cloud. In the reference implementation, these descriptions contain approximately 15%, 25%, and 60% of the original points, respectively. By combining these base descriptions, the system can generate $2^K - 1 = 7$ distinct quality levels, ranging from coarse approximations to the full-resolution point cloud.

This design differs from standard point cloud streaming, where a single representation is encoded at a fixed bitrate. Although a standard point cloud already supports viewpoint-dependent rendering at the client, it remains a monolithic representation and does not support fine-grained bitrate adaptation without re-encoding. In contrast, MDC decomposes the geometry itself into independently decodable components, allowing clients to adapt

quality by selecting subsets of already encoded data. As a result, quality adaptation is performed through *data selection* rather than *representation switching*.

The impact on encoding scalability is significant. In a conventional per-client encoding pipeline, preprocessing time increases with the number of users, as a separate representation must be generated for each client. For instance, using five encoding threads, the system sustains real-time performance (29 FPS) for 5 users, but drops to 9.6 FPS at 20 users and 5.4 FPS at 40 users due to the increased encoding load.

In contrast, the MDC approach decouples encoding from the number of users by generating a fixed number of descriptions per frame. By encoding only three descriptions, the preprocessing latency remains approximately constant (around 16 ms per frame), allowing the system to sustain real-time performance close to 30 FPS regardless of the number of connected clients.

At the delivery stage, adaptation is performed by selecting which descriptions to transmit based on the estimated bandwidth, as well as the user's field-of-view and distance to the object. Clients closer to the object or directly viewing it receive more descriptions (higher quality), while distant or out-of-view clients receive fewer or none. Because descriptions are independently decodable, this selection process does not introduce additional latency or require re-encoding.

An additional advantage of this design is graceful degradation under bandwidth constraints. Since each description contributes incrementally to the final reconstruction, partial reception still yields a valid point cloud, with quality improving as more descriptions are received. This contrasts with monolithic representations, where insufficient bandwidth leads to frame drops or severe quality switching.

Experimental results confirm that MDC stabilizes both latency and quality under increasing concurrency. The system achieves an end-to-end latency of approximately 163 ms for 3 users and 166 ms for 9 users, demonstrating that latency remains effectively constant as the number of clients increases. Moreover, while individual encoding provides higher quality for a small number of users, MDC achieves better average quality when the number of clients exceeds 20, due to its ability to maintain real-time frame rates without overloading the encoder.

By transforming encoding from a per-client operation into a shared, composable process, MDC effectively decouples preprocessing cost from user concurrency. This makes it particularly suitable for one-to-many scenarios, where scalability depends on the ability to reuse encoded data across a large number of receivers.

3.7.3 Discussion

These approaches improve scalability by eliminating per-user redundancy through different forms of sharing. MuV2 shares view-dependent outputs across users with similar viewpoints, while MDC shares encoded data across clients through composable representations. This reflects two distinct strategies for decoupling system cost from the number of users.

3.8 Transport-Level Latency Control

Transport-level mechanisms regulate how volumetric data is delivered over the network and primarily influence latency variability rather than raw latency itself. Unlike representation, encoding, or decoding strategies, which target specific stages of the volumetric pipeline, transport mechanisms do not reduce data size or computational cost. Instead, they control how data traverses the network under dynamic congestion conditions.

These challenges are not unique to volumetric systems: conventional live streaming and conferencing also depend on low queuing delay and fast congestion response. However, volumetric media amplifies the impact of transport behavior due to its higher bitrate, larger frame sizes, and lower tolerance to partial data loss. For this reason, transport mechanisms are best understood as an enabling layer that preserves the benefits of higher-level optimizations under realistic network conditions.

3.8.1 Congestion Control: L4S vs WebRTC (GCC)

In real-time volumetric video streaming, transport-layer behavior plays a critical role in determining end-to-end latency and visual stability. Unlike traditional video, volumetric streams consist of large, monolithic frames that must be received in full before decoding. As a result, both latency spikes and packet loss have a disproportionate impact on user experience, making congestion control a central component of the system.

Most existing systems rely on WebRTC, which employs Google Congestion Control (GCC). GCC is fundamentally a reactive mechanism: it estimates available bandwidth based on indirect signals such as packet loss and variations in round-trip time (RTT). These signals, however, are only observable after congestion has already occurred. When multiple flows compete for bandwidth, queues begin to build up at bottleneck links, increasing queuing delay until packets are eventually dropped. Only at that point does GCC reduce its sending rate. This delayed reaction leads to oscillatory behavior, where periods of high throughput are followed by abrupt rate reductions, resulting in unstable latency and occasional frame loss.

L4S (Low Latency, Low Loss, Scalable Throughput) [12] introduces a fundamentally different model by shifting congestion control from reactive to proactive operation. Instead of relying on packet loss or delay inflation as implicit signals, L4S leverages Accurate Explicit Congestion Notification (AccECN) to provide early and continuous feedback about network conditions. Network nodes mark packets when they detect the onset of congestion, before queues become saturated. The sender then estimates the probability of marking and adjusts its transmission rate accordingly, aiming to maintain a near-constant, sub-millisecond queuing delay.

This design effectively transforms congestion control into a closed-loop control system with fine-grained feedback. Rather than probing for available bandwidth until failure, the sender continuously tracks congestion signals and stabilizes its rate around the available capacity. As a result, queue buildup is prevented rather than corrected after the fact.

At the network level, L4S relies on a dual-queue *Active Queue Management* (AQM) architecture that separates latency-sensitive traffic from conventional flows. L4S packets are routed through a dedicated low-latency queue, where early marking replaces packet dropping as the primary congestion signal. Legacy traffic continues to use a traditional queue, ensuring backward compatibility. This separation allows L4S flows to maintain low queuing delay without being affected by bursty or loss-driven behavior from other traffic classes.

In addition to early congestion signaling, L4S enforces packet pacing at the sender. Instead of transmitting volumetric frames as bursts of packets, which would temporarily overload the network, packets are distributed over a fixed interval aligned with the frame rate. This reduces burstiness and prevents transient queue buildup, effectively shifting latency from unpredictable queuing delays to a controlled and deterministic pacing delay. As a result, transport latency becomes largely independent of instantaneous network contention.

These mechanisms have a direct impact on system performance. Experimental results show that L4S achieves significantly faster convergence of bandwidth estimation compared to GCC, reducing the time required to reach stable throughput by approximately 45%. More importantly, by reacting to early congestion signals, L4S avoids packet loss entirely under congested conditions, whereas WebRTC-based pipelines experience non-recoverable frame loss due to late retransmissions. This is particularly important for point cloud codecs, where the loss of even a small number of packets can render an entire frame undecodable.

Another key advantage of L4S is latency stability. While both L4S and WebRTC can achieve comparable average end-to-end latency, their behavior under load differs significantly. In WebRTC, latency increases with the number of competing flows due to queue buildup and slow adaptation. In contrast, L4S maintains a nearly constant transport latency, as queuing

delay is bounded and largely replaced by controlled pacing. This leads to more predictable motion-to-photon latency, which is critical for immersive and interactive applications.

Overall, L4S shifts the transport layer from a loss-driven, reactive system to a feedback-controlled, latency-oriented design. By preventing congestion rather than reacting to it, the approach enables stable low-latency streaming even under adverse network conditions, making it particularly well-suited for real-time volumetric video conferencing.

3.8.2 Rate–Utility Optimization with Window-Based Buffering

Park et al. [13] redesign the client-side control loop of volumetric streaming by jointly addressing buffer management, view-adaptive delivery, and bitrate allocation. Unlike conventional streaming systems that operate at the segment level, the proposed architecture performs fine-grained decision-making over spatio-temporal tiles, formulating streaming as a constrained *rate–utility optimization problem.*

From a pipeline perspective, the system operates at the adaptation and buffering stages. Instead of selecting a single bitrate representation per segment, the client dynamically selects both the *subset of 3D tiles* and their *level of detail (LOD)* within a sliding time window, maximizing perceived utility under a bandwidth constraint.

To support low-latency interaction, the system replaces the conventional FIFO buffer with a window-based buffer model. The buffer is defined as a time interval $[W_{\text{trail}}(t), W_{\text{lead}}(t)]$, where content at the trailing edge is consumed for rendering, while any content within the window can be requested, updated, or refined at arbitrary times. This removes the strict append-only constraint of queue-based buffers and enables two key mechanisms: (i) *insertion of newly required tiles close to playback time*, and (ii) *just-in-time quality refinement* of already buffered content. As a result, interaction latency is no longer tied to buffer length, allowing the system to maintain both responsiveness and robustness to throughput fluctuations.

The core of the system is a *rate–utility optimization framework* executed at each request opportunity. Given an estimated bandwidth budget R_i, the client selects tile representations $\mathcal{M}$ that maximize total utility

$$U(\mathcal{M}) = \sum_k U_k(m_k), \quad \text{s.t.} \quad \sum_k b_k(m_k) \leq R_i,$$

where each tile utility $U_k(m_k)$ captures its perceptual contribution. Utility is modeled as a product of three factors: (i) a bitrate-dependent quality term (logarithmic in bitrate), (ii) a level-of-detail (LOD) term that quantifies the number of distinguishable voxels based on distance and display resolution, and (iii) a visibility probability that reflects the likelihood that the tile will be within the user's viewport at playback time. This formulation

explicitly incorporates both *geometric relevance* and *user interaction uncertainty* into bitrate allocation.

The optimization problem is solved using a greedy algorithm over the convex hull of rate–utility operating points, which is provably optimal under the given formulation. Importantly, the problem decomposes across tiles, enabling efficient per-tile decisions despite the large number of candidates within the window.

Compared to conventional DASH-based approaches, which perform coarse segment-level bitrate selection, the proposed system enables fine-grained allocation of bandwidth across spatial tiles and temporal positions. Experimental evaluation demonstrates that rate–utility optimization consistently achieves higher perceived quality under the same bandwidth constraints, with double-digit improvements in aggregate utility compared to throughput- and buffer-based adaptation schemes.

The benefits are most pronounced under dynamic conditions. In multi-object scenarios, the system prioritizes tiles based on visibility and level-of-detail, leading to more efficient bandwidth usage and improved visual fidelity for relevant regions. Furthermore, the window-based buffer significantly reduces interaction latency: instead of being bounded by the full buffer duration (e.g., several seconds in queue-based systems), the system can react to viewpoint changes within a single request interval (on the order of 500 ms). This enables rapid correction of prediction errors and just-in-time refinement of visible content, resulting in more stable quality under frequent user interaction.

3.8.3 Low Latency DASH

Jansen et al. [14] investigate the feasibility of using *Low-Latency MPEG-DASH* (LL-DASH) as a transport mechanism for real-time volumetric video conferencing. Instead of relying on RTP-based protocols such as WebRTC, the system adopts an HTTP-based delivery pipeline, aiming to leverage existing web infrastructure, including CDN scalability and client-driven adaptation.

From a pipeline perspective, the approach operates at the transport and delivery stages, without modifying the volumetric representation or compression algorithms. The system encodes point cloud frames into tiled, multi-resolution representations and packages them into DASH segments, allowing the client to dynamically select tiles and quality levels based on viewpoint and bandwidth conditions.

The key enabler of low latency is the use of chunked HTTP transfer and fragmented MP4 (ISOBMFF). Instead of waiting for full segment generation, encoded data is transmitted incrementally as it becomes available. In this model, end-to-end latency is primarily deter-

mined by the *fragment duration* rather than the full segment length, enabling significantly lower delay compared to traditional DASH pipelines.

The system implements a complete end-to-end pipeline, including RGB-D capture, multi-camera fusion, point cloud tiling, voxel-based downsampling for multiple levels of detail, octree-based geometry compression, and HTTP-based segment delivery. At the client, tile selection is performed based on viewpoint and distance, enabling basic view-adaptive streaming without requiring server-side personalization.

Experimental results demonstrate that LL-DASH can achieve sub-second end-to-end latency in a volumetric conferencing setup. Specifically, measured transport latency ranges from approximately 130 ms to 450 ms, depending on point cloud size and capture configuration. The system sustains frame rates between 8–15 FPS, with compressed frame sizes ranging from 20 KB to 70 KB. These results indicate that latency is primarily dominated by data size and encoding complexity, rather than the transport protocol itself.

Compared to RTP-based pipelines, LL-DASH offers improved deployment scalability and infrastructure compatibility, as it operates over standard HTTP and can integrate with CDNs. However, its performance remains constrained by TCP transport characteristics, including head-of-line blocking and sensitivity to packet loss, which can introduce variability in delivery time.

Overall, LL-DASH demonstrates that adaptive streaming can support real-time volumetric conferencing, providing a viable alternative to custom UDP-based solutions, while exposing the limitations of HTTP/TCP transport in latency-critical interactive scenarios.

3.8.4 Discussion

Transport-level mechanisms do not reduce the fundamental cost of volumetric streaming, but regulate how data is delivered under varying network conditions. Their role is to stabilize latency and prevent network behavior from undermining optimizations at other stages of the pipeline, making them a cross-cutting layer rather than a primary source of latency reduction.

4 Comparative Analysis of Volumetric Streaming Architectures

The architectural strategies presented in Chapter 3 address latency by targeting different bottlenecks within the volumetric streaming pipeline. While each approach is effective under specific assumptions, their performance varies significantly depending on the application context.

In this chapter, we move from description to evaluation. Rather than analyzing systems in isolation, we examine their behavior under two representative use cases: large-scale event streaming and real-time volumetric conferencing. These scenarios impose different constraints on latency, scalability, and resource usage, leading to fundamentally different design requirements.

The goal of this analysis is to identify which architectural strategies are effective under each scenario, understand their limitations, and highlight the trade-offs that arise when applying them across different operating conditions.

4.1 Evaluation Framework

To enable a systematic comparison, we adopt a pipeline-oriented evaluation framework. Each system is analyzed based on the stage of the pipeline it targets—such as encoding, delivery, decoding, or transport—and the corresponding bottleneck it addresses. In addition, systems are evaluated along four key dimensions:

- **Latency:** the achievable motion-to-photon delay and its stability under varying conditions.
- **Scalability:** the ability to support increasing numbers of concurrent users without proportional increases in resource consumption.
- **Bandwidth efficiency:** the amount of data required to deliver acceptable visual quality.

- **Computational cost:** the processing requirements at the client, edge, or server, particularly under real-time constraints.

These dimensions are interdependent and often conflicting. For example, reducing bandwidth through aggressive data pruning may increase visual artifacts, while offloading computation to the edge may improve client performance but introduce additional latency. Therefore, the goal of this analysis is not to identify a single optimal solution, but to understand the trade-offs and limitations of each architectural approach under different operating conditions.

4.2 Event Streaming Systems

Large-scale volumetric event streaming involves a high number of concurrent users observing the same dynamic scene from freely selectable viewpoints. Unlike traditional video streaming, users are not passive viewers: they can navigate the scene in 6DoF, selecting arbitrary viewpoints within the environment. However, despite this freedom, user behavior remains spatially correlated, as most viewers focus on the same regions of interest (e.g., the ball or key players in a sports event).

In addition, such systems typically rely on dense multi-camera capture setups, where a large number of RGB-D cameras are deployed around the environment to enable full 6DoF navigation. While this provides spatial coverage, it significantly increases the complexity of upstream processing. The system must continuously synchronize, fuse, and compress multiple high-bandwidth streams, introducing additional latency and data volume before streaming even begins. As a result, the scalability challenge extends beyond content delivery to the entire capture-to-delivery pipeline.

This combination—*high interaction with correlated viewpoints*—defines the core challenge of event streaming. Systems must support many users simultaneously while preserving sufficient responsiveness and visual consistency. In this setting, the dominant bottlenecks are data volume, scalability, and the cost of multi-camera reconstruction, rather than strict per-user latency.

Data reduction techniques such as ViVo and saliency-driven tiling directly target the bandwidth bottleneck by transmitting only a subset of the scene. While effective in reducing data rates, these approaches rely heavily on prediction or heuristic importance estimation. This assumption becomes problematic in highly dynamic scenes such as sports events, where rapid camera motion or unexpected user behavior can lead to incorrect tile selection and visible artifacts. Moreover, their evaluation is typically limited to small-scale or controlled scenarios, leaving open the question of how they perform under realistic multi-user conditions with thousands of viewers.

Encoding-level approaches such as MDC offer a more scalable solution by decoupling encoding cost from the number of users. By enabling multiple clients to reconstruct different quality levels from a shared set of encoded descriptions, MDC avoids the linear growth in preprocessing cost observed in per-user encoding pipelines. However, this comes at the cost of reduced flexibility: since quality is constructed from predefined combinations of descriptions, the system cannot fully adapt to fine-grained spatial or perceptual differences between users. In addition, MDC does not address the underlying data volume of volumetric content, meaning that bandwidth requirements remain high even when encoding is scalable.

Cloud- and edge-rendered architectures attempt to bypass client-side limitations by offloading rendering and decoding to the server. While this reduces device-side computation, it introduces a more fundamental scalability issue: rendering cost remains tied to the number of distinct viewpoints. Even when optimized through clustering or view sharing, such systems must generate multiple view-dependent streams, causing server-side resource consumption to grow with user diversity. This makes such approaches inherently difficult to scale to the hundreds or thousands of users expected in large event streaming scenarios. Furthermore, these systems are often evaluated on small deployments (tens of users), which does not provide sufficient evidence of scalability under realistic conditions.

A common limitation across these approaches is that they address isolated bottlenecks without resolving the overall system constraints. Data reduction reduces bandwidth but introduces prediction errors, encoding optimizations improve preprocessing scalability but do not reduce transmission cost, and cloud rendering shifts computation but introduces new dependencies on network latency and server capacity. None of these approaches, in isolation, provides a complete solution for large-scale volumetric event streaming.

In addition, many systems implicitly assume powerful client devices or neglect device-level constraints. In practice, XR devices such as standalone headsets have limited processing power, thermal budgets, and battery capacity. Techniques that increase client-side workload—such as decoding large point clouds or performing continuous view-dependent processing—may lead to overheating or rapid battery drain, limiting their practical usability in extended viewing sessions.

Overall, existing approaches provide partial solutions to the event streaming problem, but their assumptions and evaluation settings often do not reflect the full complexity of real-world deployments. This indicates that event streaming cannot be supported by optimizing a single bottleneck in isolation, since bandwidth, preprocessing, and client constraints remain tightly coupled.

4.3 Conferencing Systems

Real-time volumetric conferencing differs fundamentally from event streaming in both scale and system constraints. Instead of thousands of passive viewers, conferencing involves a small number of participants who simultaneously capture, encode, transmit, and render volumetric data. This creates a *symmetric and fully interactive pipeline*, where latency is dominated not by scalability, but by the cumulative delay across encoding, transmission, and decoding stages.

In this setting, the primary requirement is maintaining a tight motion-to-photon loop, typically below 100–200 ms, while sustaining stable frame rates. Unlike event streaming, buffering cannot be used to absorb network variability without directly increasing interaction delay. As a result, the dominant bottlenecks shift from bandwidth scalability to *encoding latency*, *processing efficiency*, and *end-to-end synchronization*.

Representation-level optimizations such as GROOT address part of this problem by improving decoding performance. By restructuring octree-based representations into parallel-decodable formats, these approaches enable real-time rendering on mobile devices. However, their impact is limited by a critical asymmetry: while decoding can be accelerated through parallelization, encoding remains inherently expensive and must be performed in real time at each endpoint. In practice, encoding latency often exceeds decoding latency, making it the dominant bottleneck in the motion-to-photon pipeline. As a result, improvements in decoding alone do not significantly reduce end-to-end latency in conferencing scenarios.

Similarly, neural reconstruction approaches such as VoluSR reduce transmission bandwidth by shifting complexity to the client. While this can improve network efficiency, it introduces additional computational load at the device, which is problematic for standalone XR hardware. Continuous execution of neural models increases power consumption and thermal load, potentially leading to performance throttling or reduced session duration. In practice, this limits the applicability of such methods in sustained real-time interaction.

Transport-level optimizations, including low-latency congestion control mechanisms, contribute to stabilizing delivery but do not address the dominant bottlenecks of the conferencing pipeline. Even under ideal network conditions, latency remains constrained by encoding and processing delays. This highlights that network optimization alone is insufficient to meet the strict requirements of interactive volumetric communication.

A key limitation across many existing approaches is the assumption of asymmetric system design, where encoding is performed on powerful servers and decoding on weaker clients. While this assumption is valid for event streaming, it does not hold in conferencing scenarios, where both endpoints operate under similar hardware constraints. As a result,

the true system bottleneck shifts to real-time encoding at the edge, and techniques that rely on centralized processing or computationally expensive encoding pipelines do not translate effectively to symmetric architectures.

Furthermore, existing evaluations are often conducted under simplified conditions, such as single-user playback or offline processing, and do not fully capture the constraints of real-time bidirectional communication. In particular, few works account for the combined effects of encoding delay, device limitations, and continuous interaction over extended periods.

Overall, volumetric conferencing systems are primarily constrained by real-time processing requirements rather than scalability. Approaches that improve individual components—such as decoding or bandwidth efficiency—provide only partial benefits unless they address the full end-to-end pipeline. This suggests that practical volumetric conferencing depends primarily on reducing end-to-end processing delay under symmetric device constraints, rather than on isolated improvements in decoding or transport alone.

4.4 Cross-Use Case Comparison

The analysis of the two use cases reveals that volumetric streaming does not present a single unified problem, but rather two fundamentally different operational regimes. As a result, architectural solutions that perform well in one scenario often fail when applied to the other.

A primary source of mismatch is the *shift in dominant bottlenecks*. In event streaming, the system is constrained by bandwidth and scalability, as content must be delivered to a large number of users simultaneously. In contrast, conferencing systems are constrained by end-to-end latency, with encoding and processing delays dominating the motion-to-photon loop. This divergence directly impacts the effectiveness of existing approaches.

Data reduction techniques such as viewport-adaptive streaming are highly effective in event streaming, where reducing transmitted data directly improves scalability. However, these approaches implicitly assume predictable viewing behavior and tolerate occasional inconsistencies. In conferencing scenarios, where users interact continuously and expect consistent spatial feedback, prediction errors translate immediately into perceptual artifacts, making such methods unreliable.

Conversely, representation and decoding optimizations, such as parallel decoding architectures, are well suited for conferencing, where reducing processing latency at the client is critical. However, these approaches do not address the dominant bottleneck in event

streaming, namely the cost of delivering large volumes of data to many users. As a result, their impact remains limited when applied to large-scale scenarios.

Scalability-oriented techniques further illustrate this divergence. Methods based on shared encoding or content reuse, such as MDC, effectively eliminate per-user preprocessing overhead and scale well with the number of users. However, they achieve this by restricting flexibility in representation and adaptation, making them less suitable for interactive scenarios where per-user responsiveness is required. On the other hand, approaches that rely on per-user processing, such as cloud rendering, fundamentally conflict with scalability requirements, as resource consumption grows with the number of distinct viewpoints.

A recurring issue across many works is the reliance on assumptions that do not generalize across use cases. Systems designed for event streaming often assume asymmetric architectures with powerful servers and lightweight clients, while conferencing systems require symmetric designs where all participants operate under similar constraints. Similarly, several approaches implicitly assume that bottlenecks can be addressed independently, ignoring the strong coupling between encoding, transmission, and decoding stages.

Another limitation is the gap between evaluation and deployment conditions. Many systems are validated under small-scale or controlled environments, with limited numbers of users, simplified interaction patterns, or offline processing assumptions. These conditions do not reflect the requirements of real-world volumetric streaming, where systems must operate continuously under strict latency constraints and limited device resources. As a result, the reported performance of many approaches does not provide sufficient evidence of their practical applicability.

These observations suggest that the current body of work does not yet provide a complete solution for volumetric streaming across different use cases. Instead, existing approaches address isolated aspects of the problem, often under restrictive assumptions. Bridging this gap requires architectural designs that can adapt to different operating regimes, rather than optimizing for a single fixed scenario.‘

4.5 Design Principles for Latency-Aware Volumetric Streaming

The comparative analysis across different use cases highlights that no single architectural approach is sufficient to address the full set of challenges in volumetric streaming. Instead, effective system design requires a holistic perspective that accounts for the interaction between data volume, computation, network behavior, and application-specific constraints.

A first key principle is that *latency optimization must be aligned with the dominant bottleneck of the target application.* In large-scale event streaming, reducing data volume and enabling shared processing across users are critical for scalability. In contrast, real-time conferencing requires minimizing end-to-end processing delay, with particular emphasis on encoding efficiency and low-latency interaction. Approaches that fail to target the appropriate bottleneck often provide limited benefits, even if they optimize individual components effectively.

A second principle is that *architectural decisions introduce trade-offs across the pipeline.* Techniques that reduce bandwidth through aggressive data selection may increase the risk of visual artifacts, while shifting computation to the client or edge can lead to increased energy consumption and thermal constraints. Similarly, offloading computation to the cloud simplifies client devices but introduces additional latency and scalability challenges. As a result, latency cannot be minimized in isolation, but must be balanced against quality, robustness, and resource constraints.

A third observation is that *symmetry and deployment assumptions play a critical role in system feasibility.* Many existing approaches rely on asymmetric designs, where computationally intensive tasks are performed on powerful servers. While this is suitable for broadcast scenarios, it does not generalize to conferencing systems, where all participants operate under similar hardware constraints. Designing for symmetric execution environments requires fundamentally different strategies, particularly with respect to encoding and resource allocation.

Another important principle is that *device-level constraints must be explicitly considered.* Standalone XR devices have limited processing capability, battery capacity, and thermal headroom. Approaches that increase client-side workload—such as complex decoding pipelines or continuous neural inference—may be theoretically efficient but impractical in sustained real-world use. Ignoring these constraints leads to solutions that cannot be deployed beyond controlled experimental settings.

Finally, the analysis suggests that *future volumetric streaming systems must adopt hybrid and adaptive architectures.* Rather than relying on a fixed design, systems should dynamically combine multiple strategies—such as data reduction, scalable encoding, and transport optimization—based on current network conditions, user behavior, and device capabilities. This may involve runtime decision mechanisms that select the most appropriate pipeline configuration for a given context, potentially leveraging learning-based approaches to optimize system behavior over time.

Overall, latency-aware volumetric streaming should be viewed not as a single optimization problem, but as a multi-dimensional design space. Effective solutions will emerge from

the integration of complementary techniques, rather than from isolated improvements at individual stages of the pipeline.

5 Conclusions and Future Work

5.1 Conclusions

This thesis investigated the problem of latency in volumetric video streaming from a system-level perspective, focusing on how architectural design choices impact end-to-end performance. Rather than treating latency as a single bottleneck, the analysis showed that it emerges from the interaction of multiple components, including data generation, encoding, transmission, decoding, and rendering.

A key finding of this work is that volumetric streaming does not correspond to a single unified problem, but rather to a set of distinct operating regimes defined by the application context. In particular, the comparison between large-scale event streaming and real-time volumetric conferencing revealed a fundamental shift in dominant bottlenecks. Event streaming systems are primarily constrained by data volume and scalability, while conferencing systems are limited by encoding latency and strict end-to-end interaction requirements.

The evaluation of existing approaches demonstrated that most systems address isolated bottlenecks under specific assumptions. Data reduction techniques effectively improve bandwidth efficiency but rely on predictable user behavior. Encoding and representation optimizations improve processing performance but do not eliminate transmission cost. Cloud-based architectures offload computation but introduce scalability and latency trade-offs. As a result, no single approach is sufficient to meet the requirements of both use cases.

Furthermore, the analysis highlighted several limitations in current research. Many systems are evaluated under simplified conditions, with limited user counts or relaxed latency constraints, making it difficult to assess their applicability in real-world deployments. In addition, device-level constraints, such as processing power, thermal limits, and battery capacity, are often overlooked, despite being critical for practical XR applications.

Overall, this work demonstrates that latency-aware volumetric streaming requires a holistic approach that considers the entire pipeline and adapts to the characteristics of the target application. Effective solutions must balance bandwidth, computation, and responsiveness, rather than optimizing individual components in isolation.

5.2 Future Work

The findings of this thesis suggest several directions for future research in volumetric video streaming.

A primary direction is the development of *hybrid and adaptive architectures* that dynamically combine multiple optimization strategies based on runtime conditions. Instead of relying on a fixed pipeline, future systems should be able to adjust data reduction, encoding strategies, and transport mechanisms according to network conditions, user behavior, and device capabilities. This could be achieved through control-theoretic approaches or learning-based models that optimize system configuration in real time.

Another important direction is the improvement of *real-time encoding efficiency*. As identified in this work, encoding remains a dominant bottleneck in conferencing scenarios, particularly in symmetric systems where all participants must perform compression under strict latency constraints. Developing lightweight, parallelizable encoding techniques or hardware-accelerated solutions is essential for enabling practical real-time volumetric communication.

In addition, future research should address the challenge of *end-to-end system integration*. Many existing works focus on individual components of the pipeline, such as compression or delivery, without considering their interaction. Designing systems that jointly optimize capture, encoding, transmission, and rendering remains an open problem.

Another promising direction is the incorporation of *device-aware optimization*. Standalone XR devices impose strict constraints on power consumption and thermal behavior. Future systems should explicitly account for these limitations, adapting processing load and data delivery to maintain stable performance during prolonged usage.

Finally, there is a need for more *realistic evaluation methodologies*. Current studies are often limited to small-scale experiments or controlled environments. Future work should consider large-scale deployments, continuous operation, and realistic user interaction patterns in order to better understand system behavior under practical conditions.

Advancing these directions will be essential for enabling scalable and responsive volumetric streaming systems that can support both immersive event experiences and real-time interactive communication.

Bibliography

[1]Yongjie Guan, Xueyu Hou, Nan Wu, Bo Han, and Tao Han. "MetaStream: Live Volumetric Content Capture, Creation, Delivery, and Rendering in Real Time". en. In: *Proceedings of the 29th Annual International Conference on Mobile Computing and Networking*. Madrid Spain: ACM, Oct. 2023, pp. 1–15.

[2]Kyungjin Lee, Juheon Yi, Youngki Lee, Sunghyun Choi, and Young Min Kim. "GROOT: a real-time streaming system of high-fidelity volumetric videos". en. In: *Proceedings of the 26th Annual International Conference on Mobile Computing and Networking*. London United Kingdom: ACM, Sept. 2020, pp. 1–14.

[3]Google. *Draco: 3D Data Compression*. `https://google.github.io/draco/`. Accessed: 2026-02-16. 2017.

[4]Bo Han, Yu Liu, and Feng Qian. "ViVo: visibility-aware mobile volumetric video streaming". en. In: *Proceedings of the 26th Annual International Conference on Mobile Computing and Networking*. London United Kingdom: ACM, Apr. 2020, pp. 1–13.

[5]Jie Li, Cong Zhang, Zhi Liu, Richang Hong, and Han Hu. "Optimal Volumetric Video Streaming With Hybrid Saliency Based Tiling". en. In: *IEEE Transactions on Multimedia* 25 (2023), pp. 2939–2953.

[6]Serhan Gül, Dimitri Podborski, Jangwoo Son, et al. "Cloud rendering-based volumetric video streaming system for mixed reality services". en. In: *Proceedings of the 11th ACM Multimedia Systems Conference*. Istanbul Turkey: ACM, May 2020, pp. 357–360.

[7]Serhan Gül, Dimitri Podborski, Thomas Buchholz, Thomas Schierl, and Cornelius Hellge. "Low-latency cloud-based volumetric video streaming using head motion prediction". en. In: *Proceedings of the 30th ACM Workshop on Network and Operating Systems Support for Digital Audio and Video*. Istanbul Turkey: ACM, June 2020, pp. 27–33.

[8]Serhan Gul, Cornelius Hellge, and Peter Eisert. "Latency Compensation Through Image Warping For Remote Rendering-Based Volumetric Video Streaming". en. In: *2022 IEEE International Conference on Image Processing (ICIP)*. Bordeaux, France: IEEE, Oct. 2022, pp. 2026–2030.

[9] Anlan Zhang, Chendong Wang, Bo Han, and Feng Qian. "Efficient Volumetric Video Streaming Through Super Resolution". en. In: *Proceedings of the 22nd International Workshop on Mobile Computing Systems and Applications*. Virtual United Kingdom: ACM, Feb. 2021, pp. 106–111.

[10] Yu Liu, Puqi Zhou, Zejun Zhang, et al. "MuV2: Scaling up Multi-user Mobile Volumetric Video Streaming via Content Hybridization and Sharing". en. In: *Proceedings of the 30th Annual International Conference on Mobile Computing and Networking*. Washington D.C. DC USA: ACM, May 2024, pp. 327–341.

[11] Matthias De Fré, Jeroen Van Der Hooft, Tim Wauters, and Filip De Turck. "Scalable MDC-Based Volumetric Video Delivery for Real-Time One-to-Many WebRTC Conferencing". en. In: *Proceedings of the ACM Multimedia Systems Conference 2024 on ZZZ*. Bari Italy: ACM, Apr. 2024, pp. 121–131.

[12] Matthias De Fré, Jeroen Van Der Hooft, Chia-Yu Chang, et al. "Low-Latency Volumetric Video Conferencing in Congested Networks Through L4S". en. In: *Proceedings of the 16th ACM Multimedia Systems Conference*. Stellenbosch South Africa: ACM, Mar. 2025, pp. 113–123.

[13] Jounsup Park, Philip A. Chou, and Jenq-Neng Hwang. "Rate-Utility Optimized Streaming of Volumetric Media for Augmented Reality". en. In: *IEEE Journal on Emerging and Selected Topics in Circuits and Systems* 9.1 (Mar. 2019), pp. 149–162.

[14] Jack Jansen, Shishir Subramanyam, Romain Bouqueau, et al. "A pipeline for multiparty volumetric video conferencing: transmission of point clouds over low latency DASH". en. In: *Proceedings of the 11th ACM Multimedia Systems Conference*. Istanbul Turkey: ACM, May 2020, pp. 341–344.

List of Acronyms

2D	Two-Dimensional
3D	Three-Dimensional
360	360-Degree Video
3DoF	Three Degrees of Freedom
6DoF	Six Degrees of Freedom
AR	Augmented Reality
VR	Virtual Reality
XR	Extended Reality
RGB-D	Red-Green-Blue Depth
FoV	Field of View
OOS	Out-of-Sight
M2P	Motion-to-Photon
RTT	Round-Trip Time
FPS	Frames Per Second
QoE	Quality of Experience
GPU	Graphics Processing Unit
PD-Tree	Parallel Decodable Tree

OBB Octree Breadth Bytes

ODB Octree Depth Bytes

SR Super-Resolution

PU-GAN Point Upsampling Generative Adversarial Network

SPFH Simplified Point Feature Histogram

MDC Multiple Description Coding

GOP Group of Pictures

LOD Level of Detail

L4S Low Latency, Low Loss, Scalable Throughput

GCC Google Congestion Control

AQM Active Queue Management

AccECN Accurate Explicit Congestion Notification

DASH Dynamic Adaptive Streaming over HTTP

LL-DASH Low-Latency DASH

UDP User Datagram Protocol

TCP Transmission Control Protocol

WebRTC Web Real-Time Communication

MP4 MPEG-4 Part 14

PCC Point Cloud Compression

PRW Post-Rendering Warping

EWMA Exponentially Weighted Moving Average