https://doi.org/10.31891/2307-5732-2025-355-51
УДК 004

**MALETSKYI DENYS**
Lviv Polytechnic National University
https://orcid.org/0009-0003-1685-0472
e-mail: denys.y.maletskyi@lpnu.ua
**VYKLYUK YAROSLAV**
Lviv Polytechnic National University
https://orcid.org/0000-0003-4766-4659
e-mail: vyklyuk@ukr.net
**LI FENGPING**
Zhejiang Lab For Regenerative Medicine
https://orcid.org/0000-0002-8535-112X
e-mail: fpli@ojlab.ac.cn

# EXPLORING 3D MESH COMPRESSION: METHODS, TRADE-OFFS, AND APPLICATIONS

*Three-dimensional (3D) mesh compression has become increasingly important across a range of fields—from virtual and augmented reality to computer-aided design, gaming, and large-scale scientific visualization—where both storage constraints and real-time rendering demands continue to escalate. As 3D models grow in complexity and resolution, efficient compression becomes critical for enabling smooth interaction, fast transmission, and scalable visualization on modern hardware. This paper provides a structured overview of the primary categories of 3D mesh compression techniques: single-rate compression, progressive compression, random-accessible compression, and hybrid approaches that combine progressive refinement with selective decoding. Single-rate methods offer compact, fixed-level encodings with simple and fast decoding pipelines, though they lack adaptability in dynamic scenarios. Progressive methods construct a coarse-to-fine hierarchy, allowing for level-of-detail management and gradual refinement during streaming or transmission. Random accessible techniques emphasize localized decoding, enabling targeted access to specific mesh regions without decompressing the entire model—useful in interactive or memory-limited settings. Hybrid approaches merge these advantages, supporting both progressive reconstruction and partial access for highly responsive performance. The paper also briefly considers recent advances in neural-based compression, which offer adaptive, data-driven encoding schemes that could further enhance compression efficiency. Based on this survey, we conclude that progressive and random accessible mesh compression methods offer the most practical benefits for real-world applications that require real-time responsiveness and efficient resource usage. These approaches show strong potential and should be the focus of future research and optimization.*

***Keywords:*** *data compression, mesh compression, computer graphics.*

**МАЛЕЦЬКИЙ ДЕНИС, ВИКЛЮК ЯРОСЛАВ**
Національний університет «Львівська політехніка»
**ФЕНГПІНГ ЛІ**
Чжецзянська лабораторія регенеративної медицини

# ДОСЛІДЖЕННЯ СТИСНЕННЯ 3D-МОДЕЛЕЙ: МЕТОДИ, КОМПРОМІСИ ТА СФЕРИ ЗАСТОСУВАННЯ

*Стиснення тривимірних (3Д) моделей набуває дедалі більшого значення в різноманітних галузях — від віртуальної та доповненої реальності до автоматизованого проєктування, ігор та великомасштабної наукової візуалізації — де постійно зростають вимоги до зберігання даних і рендерингу в реальному часі. Зі збільшенням складності та роздільної здатності моделей ефективні методи стиснення стають критично важливими для забезпечення швидкої взаємодії, передачі та масштабованого візуального відображення на сучасному обладнанні. У статті подано структурований огляд основних категорій методів стиснення 3D-мешів: одноразове (single-rate) стиснення, прогресивне стиснення, стиснення з довільним доступом (random accessible), а також гібридні підходи, що поєднують прогресивне вдосконалення з вибірковим декодуванням. Одноразові методи забезпечують компактне кодування з фіксованим рівнем деталізації та швидким декодуванням, однак обмежені в гнучкості. Прогресивні методи формують ієрархію від грубого до точного відтворення, що дозволяє керувати рівнями деталізації та поступовим завантаженням. Методи з довільним доступом орієнтовані на часткове декодування, що дає змогу обробляти лише потрібні частини моделі без завантаження її повністю — це особливо корисно у взаємодіючих або обмежених за ресурсами середовищах. Гібридні підходи об'єднують ці переваги, забезпечуючи як поступове відновлення, так і вибірковий доступ для досягнення високої адаптивності в режимі реального часу. У статті також коротко розглянуто новітні досягнення у сфері стиснення, заснованого на нейронних мережах, які відкривають перспективи адаптивного, орієнтованого на дані кодування. На основі проведеного аналізу зроблено висновок, що прогресивне стиснення та методи з довільним доступом мають найбільші практичні переваги для реальних завдань, які потребують інтерактивності та ефективного використання ресурсів. Саме ці підходи слід розглядати як пріоритетні напрями подальших досліджень і вдосконалення.*

***Ключові слова:*** *стиснення данних, стиснення моделей, комп'ютерна графіка.*

Стаття надійшла до редакції / Received 27.05.2025
Прийнята до друку / Accepted 26.06.2025

## Introduction

Three-dimensional (3D) meshes serve as a fundamental representation for digital geometry across numerous industries, from entertainment and virtual reality to engineering and medical imaging. As these applications continue to grow and demand ever larger, more detailed models, efficient compression methods have become critical to store,

transmit, and render these massive datasets. High compression ratios can drastically reduce bandwidth and storage requirements, yet preserving critical geometric and topological details remains imperative, particularly for domains such as CAD or healthcare, where inaccuracies can compromise downstream tasks.

Over the past several decades, researchers have advanced a variety of strategies to tackle the unique challenges of 3D mesh compression. Early techniques focused on reordering and quantizing mesh data, whereas more sophisticated algorithms introduced predictor-based and transform-domain methods, achieving higher compression ratios at increasingly manageable error levels. Progressive and random-accessible approaches further enable selective or incremental loading and refinement of complex meshes, striking a balance between efficient data storage and real-time responsiveness. As the field moves forward, hybrid solutions and hardware-friendly formats—often leveraging novel data structures or even neural networks—continue to push the boundaries of mesh compression, aiming for scalable performance without compromising on fidelity or flexibility. This paper surveys a wide range of these approaches, evaluating their compression rates, quality retention, and decompression times across different use cases and constraints.

## Overview of 3D Mesh Compression

Mesh compression seeks to encode the vertices, faces, and connectivity of a 3D object in as few bits as possible, while retaining sufficient fidelity for the target application. This is crucial in areas such as gaming, computer-aided design, and medical visualization, where high-polygon models can quickly exceed memory and bandwidth limits. At its core, a compressed mesh stores information about geometry (vertex positions) and topology (connectivity among vertices and faces), with optional attributes like normals, textures, or per-vertex colors also subject to compression.

Over the years, a variety of compression paradigms have emerged, each trading off between storage efficiency, reconstruction quality, and the ability to decode portions of the data on demand. In this article, we divide the discussion into four major categories. Single-rate mesh compression techniques (Section 3.1) deliver a one-shot encoding of the mesh at a fixed resolution and are often lauded for simplicity and strong baseline ratios. Progressive Meshes Compression (Section 3.2) builds a coarse-to-fine representation, allowing real-time level-of-detail (LOD) control. Random Accessible Mesh Compression (Section 3.3) emphasizes localized or partial decoding, enabling on-demand retrieval of specific regions without decompressing the entire model. Finally, progressive and random accessible mesh compression approaches (Section 3.4) unify these objectives, permitting coarse-to-fine refinement while also supporting region-based access. Each category tackles different practical requirements, from storing massive static models to seamlessly streaming geometry in real-time interactive applications.

## Methods for Compression

### 1. Single-rate mesh compression.

One of the earliest strategies to reduce 3D mesh storage while preserving exact connectivity and geometry involves reorganizing mesh triangles into “strips.” A triangle strip is a sequential arrangement of connected triangles that share edges, so each new triangle (after the initial one) is specified by just one new vertex. This concept reduces redundancy by omitting repeated edge references [1], [2], [3]. In parallel, a straightforward way to lower geometric precision without fundamentally altering connectivity is to quantize vertex coordinates. Although often a slightly “lossy” step (due to rounding), coarse quantization can be avoided by choosing a sufficiently high bit resolution. In practice, 16 bits or 24 bits per coordinate can maintain full visual fidelity in many applications, thus serving effectively as a near-lossless or visually lossless strategy [4], [5].

Building on these ideas, Michael Deering proposed more flexible schemes, often called generalized triangle strips or generalized triangle meshes [3]. Rather than requiring strictly contiguous strips of triangles, these methods accommodate broader connectivity patterns with fewer state changes. By minimizing repeated indexing for shared edges and vertices, they can further reduce storage overhead. A subsequent leap in lossless geometry compression came from predictor-based algorithms, which encode each new vertex by predicting its position from already decoded neighbors and then compress the difference rather than storing absolute coordinates. Early predictive methods by Gabriel Taubin and Jarek Rossignac demonstrated how local neighborhood information can significantly cut redundancy in vertex data, while Touma and Gotsman introduced the parallelogram predictor [6] to estimate a vertex’s coordinates from two adjacent edges. Such residual-based coding preserves precise geometry as long as prediction errors are stored exactly.

Connectivity—the graph describing how vertices and faces connect—also benefits from specialized compression. Edgebreaker [7] systematically “peels” faces and labels each triangle with a simple symbol (C, L, E, R, S) to reduce topological data. Valence-driven connectivity [8] leverages the fact that most meshes exhibit a predictable distribution of vertex valences, achieving high compression ratios without sacrificing any mesh structure. Beyond these strategies, several influential methods operate in a transform domain or use space-partitioning. For instance, Karni and Gotsman [9] transform vertex coordinates into the mesh’s frequency domain via the graph Laplacian; high-frequency components can be truncated or quantized, yielding more compact data. Guskov and Khodakovsky [10] similarly employ wavelet transforms to exploit temporal and spatial coherence, encoding both geometry and connectivity with significant space savings. Lengyel et al. [11] extend spectral methods by reusing a fixed set of basis functions, speeding up certain computations while retaining many benefits (and computational costs) of frequency-based approaches.

For extremely large models, the CHuMI Viewer ("Compressive Huge Mesh Interactive Viewer") [12] uses a kd-tree (or nSP-tree) to partition coordinate space. This organization allows localized geometry storage and precision adaptation per region, facilitating interactive streaming for meshes with hundreds of millions of polygons. New connectivity-centric methods continue to appear, such as the triangular matrix-based lossless compression algorithm [13], which arranges faces in a matrix structure to exploit planar graph properties, reducing the bits required for storing complex connectivity, though geometry typically still relies on additional compression techniques. Meanwhile, dynamic meshes—where geometry or connectivity changes over time—can also be compressed using hierarchical displacement encoding. Song, Kang, and Jung in "Hierarchical Arithmetic Coding of Displacements for Dynamic Mesh Compression" [14] describe a method that refines vertex positions at each level with small displacements, then entropy-codes these displacements via arithmetic coding. By focusing on incremental updates to a stable base configuration, it efficiently stores per-frame offsets and can be extended to near-lossless or lossy compression while preserving accurate vertex motion.

**2. Progressive Meshes Compression**

A key challenge when rendering large-scale 3D models is efficiently transmitting and processing geometry at different levels of detail (LOD). Progressive meshes address this by delivering a coarse-to-fine representation, so a client or renderer starts with a simplified mesh and refines it incrementally only as needed—an advantage for distance culling or real-time LOD adjustments. Hoppe's seminal work, "Progressive Meshes" [15], introduced a scheme that begins with a heavily simplified base mesh and reintroduces detail through vertex-split operations. Each split restores collapsed vertices and faces, yielding a continuous LOD pipeline with manageable storage. He later refined this method in "View-Dependent Refinement of Progressive Meshes" [16], targeting only camera-facing or visually significant regions for higher resolution, thereby preserving performance by leaving other areas coarser.

Taubin, Gueziec, Horn, and Lazarus proposed Progressive Forest Split Compression [17] to reorganize a sequence of split operations into a "forest" of spanning trees. Each tree references a sub-region of the mesh, revealing partial structure at each refinement step. This forest-based method efficiently encodes both geometry and connectivity changes while maintaining the key benefits of progressive retrieval. Pajarola and Rossignac's Compressed Progressive Meshes [18] further build on Hoppe's concept, adding batching of vertex splits and sophisticated geometry predictors (e.g., butterfly schemes) to push compression ratios higher. By separately storing geometry deltas from the connectivity stream, they also enable distinct encoding strategies tailored to each data type.

Another thread of research involves wavelet transforms, as demonstrated by Valette and Prost in "A Wavelet-Based Progressive Compression Scheme For Triangle Meshes: Wavemesh"[19]. By decomposing the surface into frequency bands, they send high-frequency details last or omit them at ithe nitial stages. This approach offers flexibility in managing bandwidth, as users can stop decoding at an acceptable quality level or continue until the mesh is fully refined. Meanwhile, Progressive Lossless Mesh Compression via Incremental Parametric Refinement [20] guarantees exact reconstruction without geometric or topological errors by incrementally refining a parametric domain mapped to the mesh. Although more computationally demanding than simpler progressive schemes, it ensures that each newly introduced vertex or face accurately matches the original model, preserving an entirely lossless progression.

**3. Random Accessible Mesh Compression**

Random accessible mesh compression techniques prioritize partial or on-demand decoding of only those regions needed at any given time. Streaming Meshes [21] initially demonstrated how reorganizing faces and vertices into small, sequentially processed chunks reduces cache misses and avoids loading the entire mesh into memory. Building on this idea, Streaming Compression of Triangle Meshes [22] applies compression within each chunk, creating independently decodable stream units and thereby supporting interactive visualization of specific regions based on view or priority.

Other approaches strengthen random access by introducing explicit indexing for each compressed data block, as in Random-Accessible Compressed Triangle Meshes [23], ensuring near-constant time decompression of faces upon request. Mesh Chartification [24] pushes this further by dividing the model into independently compressible "charts." Each chart maintains localized connectivity and boundary information through a base wire mesh, so any specific portion can be decoded on-demand with little overhead—an asset for large-scale or distributed rendering. An emerging direction is meshlet-based compression, highlighted by "Towards Practical Meshlet Compression" [25], which encodes small, GPU-friendly clusters of faces (meshlets). Each meshlet can be selectively loaded and decompressed, aligning well with modern rendering pipelines' culling and shading strategies. Finally, Neural Geometry Fields for Meshes [26] adopt a learning-based approach, training a neural network to reconstruct geometry from compact latent representations. Although inference is more computationally intensive, it can capture shared features across multiple shapes, potentially enabling partial reconstruction or advanced editing and style transfers.

**4. Progressive and random accessible mesh compression**

While progressive mesh algorithms primarily focus on coarse-to-fine refinements and random accessible schemes emphasize localized decoding, there are hybrid approaches designed to merge both capabilities. By doing so, they allow on-demand loading and viewing at multiple levels of detail while also supporting selective decompression of specific regions. This combination is especially beneficial for large-scale or distributed 3D pipelines where it is crucial to minimize memory usage, data transfer, and rendering latency.

A representative method, POMAR (Compression of Progressive Oriented Meshes Accessible Randomly) [27], enables stepwise refinements while preserving random access to any region at a chosen LOD. It tracks oriented patches and their subdivision history, encoding global connectivity updates alongside local geometry changes in a manner that supports immediate retrieval of coarse overviews or selective refinement of high-detail areas. This dual capability is vital for real-time visualization, where resources are limited or user-driven exploration requires rapid focus on different parts of the model.

Multiresolution Random Accessible Mesh Compression [28] pursues similar goals by coupling progressive-like encoding with random-access chunking. Each chunk encapsulates localized geometry and connectivity details, letting interactive systems load only the relevant segments at higher resolutions. Although this hierarchical organization imposes overhead, it can markedly improve real-time performance for large models.

Additional research continues to refine these dual strategies. For example, Balsa Rodriguez, Gobbetti, and Pajarola [29] describe a compression-domain random-access framework integrating view-dependent rendering with partial loading of polygonal data. Collectively, these techniques strive to reduce data footprints and decoding costs without sacrificing either the flexibility of refined detail or the responsiveness of targeted decompression—key requirements in large-scale 3D visualization and interactive graphics.

**Comparative Analysis**

Below is a two-part examination of the methods presented in Sections 3.1 through 3.4, focusing on three key evaluation metrics - compression ratio, quality retention, and computational efficiency - followed by a comparison table summarizing their performance characteristics. We conclude with a discussion of the main trade-offs and considerations when choosing among these methods.

**1. Evaluation Metrics**

Assessing 3D mesh compression typically involves quantifiable metrics that capture compression ratio, quality retention, and computational efficiency. Below are representative formulas illustrating how these metrics are often expressed.

**Compression Ratio**

A fundamental measure is the compression ratio, which indicates the factor by which data is reduced. Let Soriginal be the size of the uncompressed data (in bits or bytes) and Scompressed be the size of the compressed data. Then the compression ratio R can be defined as:

$$R = \frac{S_{original}}{S_{compressed}}, \quad (1)$$

Alternatively, in bits per vertex (bpv) notation, one may track:

$$bpv = \frac{S_{compressed}}{n}, \quad (2)$$

where n is the number of vertices. Traditional geometry compression can yield tens of bpv for the geometry component (after quantization and entropy coding), while connectivity can reach single-digit bpv. Predictive or transform-based methods often push these numbers lower, depending on mesh smoothness, connectivity complexity, and whether partial or full decompression is required.

**Quality Retention**

Quality or distortion measures how closely the reconstructed mesh approximates the original. One common metric is the root mean square (RMS) error between corresponding vertex positions:

$$RMS = \sqrt{\frac{1}{n}\sum_{i=1}^{n} \left|\left|v_i - \tilde{v_i}\right|\right|^2}, \quad (3)$$

where $v_i$ is the original vertex position and $\tilde{v_i}$ is the decoded vertex.

**Computational Efficiency**

Practical usage often depends on measuring the time required to compress and decompress a 3D mesh, as well as the memory resources involved. Compression time reflects how long it takes to reduce a model to its compressed form, whereas decompression time indicates how quickly that model can be reconstructed for rendering or processing. Both metrics depend on factors such as algorithmic complexity, implementation details, and the degree of hardware parallelism or acceleration available.

Memory usage also plays a significant role. Some methods, particularly those based on blocks or charts, can process only a portion of the mesh in memory at any one time, thereby reducing overall system demands. Approaches that enable on-demand loading, such as streaming or progressive schemes, can further lower peak memory requirements by unloading data that is no longer needed. In the end, an ideal compression algorithm balances a high reduction in data size with short encode and decode times and minimal memory consumption, especially for large-scale or real-time 3D environments.

**2. Comparison Table**

Below is a concise, qualitative table comparing representative methods from single-rate, progressive, and random accessible categories. Each category's entries capture the approach's typical performance with respect to our three metrics.

Table 1

**Comparison table of various mesh compression techniques**

| Approach | Approx. Compression Rate (bpv or ratio) | Quality | Decompression Time |
|---|---|---|---|
| Triangle Strips & Vertex Quantization (Clark; Foley et al.; Deering) | ~10–30 bpv total (geometry + connectivity) | Near-lossless if 16–24 bits per coordinate. (some rounding) | Low (simple indexing + coordinate scaling) |
| Generalized Triangle Strips (Deering) | Similar or slightly better than basic strips (~8–25 bpv) | Near-lossless with adequate quantization | Low–Moderate (managing larger vertex buffer) |
| Predictor-Based (Taubin, Rossignac; Touma & Gotsman) | Can drop to ~5–20 bpv, depending on the mesh | Lossless if no extra quantization (storing residuals exactly) | Low–Moderate (predictor decoding + entropy) |
| Edgebreaker (Rossignac) | Connectivity ~2–4 bits/triangle; geometry separate, total ~8–25 bpv | Lossless topology (no distortion in connectivity) | Low (straightforward symbol decoding) |
| Valence-Driven Connectivity (Alliez & Desbrun) | Connectivity ~1–3 bits/triangle; geometry separate, total ~8–25 bpv | Lossless connectivity | Low–Moderate (valence-based reconstruction) |
| Spectral Mesh Compression (Karni & Gotsman) | ~10–40 bpv (varies with frequency truncation) | Tunable from near-lossless to lossy (dropping high-frequency bands) | Moderate–High (inverse spectral transform) |
| Wavelet (Parametric Sequences) (Guskov & Khodakovsky) | ~15–50 bpv (coherent animations can see higher gains) | Tunable; can discard higher wavelet coefficients | Moderate–High (multi-level wavelet decoding) |
| 3D Mesh Compression <br>Using Fixed Spectral Bases (Lengyel et al.) | ~10–30 bpv (similar to general spectral) | Adjustable—can truncate high-frequency components | Moderate–High (fixed basis reconstruction) |
| CHuMI Viewer (Local kd-tree Partitioning) | ~10–40 bpv overall (massive models) | Usually near-lossless (adaptive bit precision) | Moderate (block-based decode + precision checks) |
| Triangular Matrix-Based Lossless (He et al.) | Connectivity <5–10 bpv, geometry separate, total ~14–35 bpv | Lossless connectivity | Low–Moderate (matrix traversal and vertex binding) |
| Hierarchical Arithmetic Coding (Song et al.) (Dynamic Displacements) | ~10–30 bpv per frame | Near-lossless or lossy (depends on quantization) | Moderate (arithmetic decoding of displacements) |
| Hoppe's Progressive Meshes | ~8–20 bpv once fully refined | Near-lossless or visually lossless | Low–Moderate (vertex-split decoding) |
| View-Dependent Refinement (Hoppe) | ~8–20 bpv, but partial refinement | Region-specific fidelity | Moderate (dynamic LOD logic) |
| Progressive Forest Split (Taubin et al.) | ~10–25 bpv | Lossless connectivity; geometry can be near-lossless | Low–Moderate (decoding forest updates) |
| Compressed Progressive Meshes (Pajarola & Rossignac) | ~15–30 bpv | Near-lossless if quantization is fine | Moderate (batch processing + predictor decode) |
| Wavelet-Based Progressive (Alliez & Desbrun) | ~15–40 bpv | Tunable; can omit high-frequency detail | Moderate–High (wavelet-based refinement) |
| Progressive Parametric (Ibarria et al.) | ~8–20 bpv (fully lossless) | Exactly lossless (no geometric errors) | High (incremental parametric domain building) |
| Streaming Meshes (Isenburg & Lindstrom) | ~10–25 bpv (with optional compression) | Can be lossless if geometry is unquantized | Low–Moderate (chunk-based streaming decode) |
| Streaming Compression of<br>Triangle Meshes | ~15–30 bpv | Near-lossless if quantization is modest | Moderate (chunk-level decoding overhead) |

Table 1 continuation

| Approach | Approx. Compression Rate (bpv or ratio) | Quality | Decompression Time |
|---|---|---|---|
| Random-Accessible Compressed<br>Triangle Meshes | ~10–25 bpv | Usually near-lossless (geometry can be quantized) | Moderate (indexing overhead for random access) |
| Mesh Chartification | ~8–25 bpv | Typically lossless within each chart | Low–Moderate (boundary references) |
| Meshlet Compression ("Towards Practical…") | ~15–35 bpv | Near-lossless with quantized coordinates | Moderate (meshlet structure decoding) |
| Neural Geometry Fields (Liu et al.) | ~20–100+ bpv (can be extremely compact if multi-shape) | Lossy; depends on the network inference accuracy | High (run-time neural inference) |
| POMAR (Progressive + Random Access) | ~12–30 bpv | Near-lossless if residuals stored exactly | Moderate–High (oriented patch decode + partial refine) |
| Multiresolution Random Accessible (Szymczak & Rossignac) | ~10–30 bpv | Near-lossless (geometry can be quantized) | Moderate (hierarchical chunk decode) |
| Balsa Rodriguez et al. (Random-Access & Progressive) | ~8–25 bpv (varies by single-rate vs. progressive) | Tunable fidelity, partial or full LOD | Moderate (indexing + LOD decode) |

## Discussion

**1. Trade-offs for Single-Rate vs. Progressive.**

Single-rate methods (e.g., triangle strips, generalized strips, predictor-based encoding, Edgebreaker) generally deliver a one-shot reconstruction at a specific resolution. They excel in scenarios where the model is fully downloaded or stored once and then rendered, offering straightforward implementations and high compression ratios. Progressive approaches, by contrast, introduce a hierarchy of mesh representations. Although they often incur a slight increase in storage overhead (due to metadata for splits or transformations), they enable level-of-detail (LOD) control and more efficient streaming in contexts where partial or incremental viewing is desired.

**2. Random Accessibility Benefits.**

Random-accessible schemes partition the mesh into small, independently decodable units, making them suitable for large models or applications where only parts of the mesh need to be viewed or edited at a time. This partitioning, however, necessitates additional data structures (e.g., chunk boundaries, indexing tables, or chart cross-references) that may slightly reduce overall compression efficiency or increase encoding/decoding complexity. In return, they offer crucial benefits for out-of-core rendering pipelines, distributed computing environments, and any interactive application that requires on-demand retrieval of geometry.

**3. Combining Progressive and Random Accessibility.**

When both progressive refinement and local decoding are required, methods like POMAR balance the ability to selectively load details with the option to refine the entire mesh progressively. While these "hybrid" approaches can involve the most complex data management (keeping track of partial LOD states, oriented patches, boundary references, etc.), they allow flexible rendering in networked or resource-limited scenarios. The overhead from storing both progressive and random accessible metadata is outweighed by the advantage of being able to retrieve specific regions at different resolutions with minimal delay.

**4. Choosing the Right Technique.**

In practice, selecting a mesh compression algorithm depends on the target application, available bandwidth, and computational resources. High-end engineering or medical contexts might favor near-lossless or lossless approaches (predictor-based or specialized connectivity encoding) to preserve fidelity. Real-time rendering of large-scale scenes may benefit more from random-accessible or progressive methods. Hybrid solutions should be considered if both partial decoding and multiple LODs are integral to the workflow. Ultimately, understanding the interplay of compression ratio, rendering performance, and implementation complexity is essential for choosing the best-suited solution.

## Applications and Future Trends

Progressive and random accessible mesh compression techniques hold significant promise for modern rendering pipelines, where large datasets need to be visualized in real time across diverse platforms. By offering on-demand access to specific regions of a model at varied levels of detail, these hybrid methods can efficiently accommodate both local culling (e.g., dynamic frustum or distance-based) and global streaming. The random accessibility component allows high parallelization on GPUs, as different segments of the mesh can be decoded independently; this also lends itself to more effective load balancing and better exploitation of hardware resources.

Meanwhile, the progressive aspect delivers real-time LOD adjustments that maintain performance even as scene complexity and camera movements evolve.

Beyond real-time rendering, fields such as CAD, GIS, and medical imaging can benefit from partial-decoding strategies to interact with large-scale or detailed models on constrained devices or networks. However, further research is needed to optimize hybrid schemes that merge random-access chunking with multiresolution geometry encoding, aiming to reduce overheads while preserving high fidelity. Neural network–based compression also remains an exciting frontier: if the bottleneck of inference-heavy decompression can be mitigated or hybridized with more traditional decoding techniques, these methods could offer exceptional compression ratios without sacrificing interactive performance. By embedding neural fields or latent representations into a conventional mesh pipeline, developers might achieve the best of both worlds: high compression rates from learned encodings and swift decompression through GPU-friendly data structures or partial expansions. As hardware accelerators for AI continue to advance, such solutions may soon become practical for a wide range of 3D applications, enabling sophisticated geometry compression that is both scalable and fully responsive to real-time demands.

## Conclusion

3D mesh compression has matured into a key enabler of interactive visualization, resource-efficient storage, and high-fidelity rendering. As models grow in polygon count to capture intricate details demanded by applications such as virtual reality, product design, and remote collaboration, the pressure on both data transmission and real-time rendering pipelines intensifies. In response, researchers have developed a spectrum of methods ranging from straightforward single-rate compression, where a fixed, highly optimized encoding is generated for a particular resolution, to elaborate, multi-stage processes that allow selective, progressive, or random-access decoding. Each technique trades off distinct factors: compression ratio, computational overhead, memory usage, and the final quality of reconstruction.

Single-rate solutions often excel at providing strong baseline ratios while maintaining relatively simple decode paths, making them suitable for static environments or offline scenarios. Progressive approaches, by contrast, supply a smooth path from coarse mesh versions to full detail, supporting real-time level-of-detail (LOD) adjustments that help keep rendering workloads manageable. Random accessible schemes address the need to retrieve only subsets of a model, a boon for limited-memory or bandwidth-constrained scenarios and for large-scale virtual environments where only certain sections of geometry are relevant at any given moment. When combined, progressive and random accessible methods can unlock unprecedented flexibility, letting users not only refine in stages but also decode precisely the regions of interest in high detail, all without a full model load.

Amid these established paradigms, the integration of neural network–based compression represents a burgeoning frontier. Such strategies can yield remarkable data reductions by leveraging learned feature representations, particularly when multiple meshes share structural similarities. However, practical deployment hinges on lowering the decoding latency inherent in neural inference. Future research may converge on hybrid methods that store latent codes in a neural format while capitalizing on proven, GPU-friendly partial decoding techniques. By optimizing this balance, developers may achieve the twin goals of deep compression and real-time responsiveness.

Looking ahead, mesh compression is poised to remain a vital research area. Industrial and academic applications will demand ever more efficient encodings—both to fit the constraints of mobile devices, web platforms, or immersive headsets, and to streamline cross-platform content creation pipelines. Additionally, with the anticipated growth of metaverse-like experiences and volumetric capture, new compression challenges will emerge, pushing existing algorithms to adapt or inspiring wholly novel solutions. While no single scheme can claim universal dominance, the collective body of techniques discussed here—single-rate, progressive, random accessible, and hybrid—will serve as a robust toolkit, guiding users toward the approach best suited to their fidelity requirements, performance objectives, and evolving hardware capabilities.

## References

1. Clark J.H. Hierarchical geometric models for visible-surface algorithms / J.H. Clark // SIGGRAPH ’76. – 1976. – P. 267.

2. Land R., Foley J.D., Dam A.V. Fundamentals of Interactive Computer Graphics / R. Land, J.D. Foley, A.V. Dam // Leonardo. – 1984. – P. 59.

3. Deering M. Geometry compression / M. Deering // Proceedings of SIGGRAPH ’95. – ACM Press, 1995. – P. 13–20.

4. Chow M.M. Optimized geometry compression for real-time rendering / M.M. Chow // Proceedings Visualization ’97. – 1997. – P. 347–354.

5. Rossignac J. Edgebreaker: connectivity compression for triangle meshes / J. Rossignac // IEEE Transactions on Visualization and Computer Graphics. – 1999. – Vol. 5, № 1. – P. 47–61.

6. Alliez P., Desbrun M. Valence-Driven Connectivity Encoding for 3D Meshes / P. Alliez, M. Desbrun // Computer Graphics Forum. – 2001. – Vol. 20, № 3. – P. 480–489.

7. Guskov I., Khodakovsky A. Wavelet compression of parametrically coherent mesh sequences / I. Guskov, A. Khodakovsky // Proceedings of SCA ’04. – 2004. – P. 183.

8. Valette S., Prost R. A Wavelet-Based Progressive Compression Scheme For Triangle Meshes: Wavemesh / S. Valette, R. Prost // IEEE Transactions on Visualization and Computer Graphics. – 2004. – Vol. 10. – P. 123–129.
9. Isenburg M., Lindstrom P. Streaming meshes / M. Isenburg, P. Lindstrom. – 2005. – P. 238.
10. Maglo A., Grimstead I., Hudelot C. POMAR: Compression of progressive oriented meshes accessible randomly / A. Maglo, I. Grimstead, C. Hudelot // Computers & Graphics. – 2013. – Vol. 37, № 6. – P. 743–752.
11. Karni Z., Gotsman C. 3D Mesh Compression Using Fixed Spectral Bases / Z. Karni, C. Gotsman. – [Без вихідних даних].
12. CHuMI Viewer: Compressive Huge Mesh Interactive Viewer. – 2024. – ResearchGate. – DOI: 10.1016/j.cag.2009.03.029.
13. Balreira D., da Silveira T. Triangular matrix-based lossless compression algorithm for 3D mesh connectivity / D. Balreira, T. da Silveira // The Visual Computer. – 2024. – Vol. 40. – DOI: 10.1007/s00371-024-03400-8.
14. Nishimura H., Kato H., Kawamura K. Hierarchical Arithmetic Coding of Displacements for Dynamic Mesh Compression / H. Nishimura, H. Kato, K. Kawamura. – 2023. – P. 2854. – DOI: 10.1109/ICIP49359.2023.10222117.
15. Hoppe H. Progressive meshes / H. Hoppe. – [Без вихідних даних].
16. Hoppe H. View-dependent refinement of progressive meshes / H. Hoppe // SIGGRAPH ’97. – ACM Press, 1997. – P. 189–198.
17. Progressive Forest Split Compression. – ResearchGate. – Accessed: Jan. 23, 2025. – [Online]. Available: https://www.researchgate.net/publication/2497716_Progressive_Forest_Split_Compression
18. Pajarola R., Rossignac J. Compressed Progressive Meshes / R. Pajarola, J. Rossignac // IEEE Transactions on Visualization and Computer Graphics. – 2000. – Vol. 6. – P. 79–93.
19. Valette S., Prost R. A Wavelet-Based Progressive Compression Scheme For Triangle Meshes: Wavemesh / S. Valette, R. Prost // IEEE Transactions on Visualization and Computer Graphics. – 2004. – Vol. 10. – P. 123–129.
20. Valette S., Chaine R., Prost R. Progressive Lossless Mesh Compression Via Incremental Parametric Refinement / S. Valette, R. Chaine, R. Prost. – 2009. – [Online]. Available: https://doi.org/10.1111/j.1467-8659.2009.01507.x
21. Isenburg M., Lindstrom P. Streaming meshes / M. Isenburg, P. Lindstrom. – 2005. – P. 238. – DOI: 10.1109/VISUAL.2005.1532800.
22. Isenburg M., Lindstrom P., Snoeyink J. Streaming compression of triangle meshes / M. Isenburg, P. Lindstrom, J. Snoeyink // SIGGRAPH ’05. – ACM Press, 2005. – P. 136.
23. Random-Accessible Compressed Triangle Meshes. – ResearchGate. – Oct. 2024. – DOI: 10.1109/TVCG.2007.70585.
24. Random Accessible Mesh Compression Using Mesh Chartification. – ResearchGate. – Accessed: Jan. 28, 2025. – [Online]. Available: https://www.researchgate.net/publication/23472936_Random_Accessible_Mesh_Compression_Using_Mesh_Chartification
25. Kuth B. et al. Towards Practical Meshlet Compression / B. Kuth et al. – 2024. – arXiv:2404.06359. – [Online]. Available: http://arxiv.org/abs/2404.06359
26. Edavamadathil Sivaram V., Li T.-M., Ramamoorthi R. Neural Geometry Fields For Meshes / V. Edavamadathil Sivaram, T.-M. Li, R. Ramamoorthi // SIGGRAPH Conference Papers ’24. – ACM, 2024. – P. 1–11.
27. Maglo A., Grimstead I., Hudelot C. POMAR: Compression of progressive oriented meshes accessible randomly / A. Maglo, I. Grimstead, C. Hudelot // Computers & Graphics. – 2013. – Vol. 37, № 6. – P. 743–752.
28. Kim J., Choe S., Lee S. Multiresolution Random Accessible Mesh Compression / J. Kim, S. Choe, S. Lee // Computer Graphics Forum. – 2006. – Vol. 25, № 3. – P. 323–331.
29. Balsa Rodríguez M. et al. State-of-the-Art in Compressed GPU-Based Direct Volume Rendering / M. Balsa Rodríguez et al. // Computer Graphics Forum. – 2014. – Vol. 33, № 6. – P. 77–100.