HexaShrink, an exact scalable framework for hexahedral meshes with attributes and discontinuities: multiresolution rendering and storage of geoscience models

Jean-Luc Peyrot, Laurent Duval, Frédéric Payan, Lauriane Bouard, Lenaic Chizat, Sébastien Schneider, Marc Antonini

HexaShrink, an exact scalable framework for hexahedral meshes with attributes and discontinuities: multiresolution rendering and storage of geoscience models

Jean-Luc Peyrot, Laurent Duval, Frédéric Payan, Lauriane Bouard, Lenaic Chizat, Sébastien Schneider, Marc Antonini

Computational Geosciences (2019) 23:723–743
https://doi.org/10.1007/s10596-019-9816-2

ORIGINAL PAPER

HexaShrink, an exact scalable framework for hexahedral meshes with attributes and discontinuities: multiresolution rendering and storage of geoscience models

Jean-Luc Peyrot[1] · Laurent Duval[2,3] · Frédéric Payan[4] · Lauriane Bouard[1] · Lénaïc Chizat[2,5] · Sébastien Schneider[1,6] · Marc Antonini[4]

Received: 26 July 2018 / Accepted: 20 March 2019 / Published online: 3 May 2019

Abstract

With huge data acquisition progresses realized in the past decades and acquisition systems now able to produce high resolution grids and point clouds, the digitization of physical terrains becomes increasingly more precise. Such extreme quantities of generated and modeled data greatly impact computational performances on many levels of high-performance computing (HPC): storage media, memory requirements, transfer capability, and finally simulation interactivity, necessary to exploit this instance of big data. Efficient representations and storage are thus becoming "enabling technologies" in HPC experimental and simulation science. We propose HexaShrink, an original decomposition scheme for structured hexahedral volume meshes. The latter are used for instance in biomedical engineering, materials science, or geosciences. HexaShrink provides a comprehensive framework allowing efficient mesh visualization and storage. Its exactly reversible multiresolution decomposition yields a hierarchy of meshes of increasing levels of details, in terms of either geometry, continuous or categorical properties of cells. Starting with an overview of volume meshes compression techniques, our contribution blends coherently different multiresolution wavelet schemes in different dimensions. It results in a global framework preserving discontinuities (faults) across scales, implemented as a fully reversible upscaling at different resolutions. Experimental results are provided on meshes of varying size and complexity. They emphasize the consistency of the proposed representation, in terms of visualization, attribute downsampling and distribution at different resolutions. Finally, HexaShrink yields gains in storage space when combined to lossless compression techniques.

Keywords Compression · Corner point grid · Discrete wavelet transform · Geometrical discontinuities · Hexahedral volume meshes · High-performance computing · Multiscale methods · Simulation · Upscaling

1 Introduction

Simulation sciences and scientific modelling in high-performance computing employ meshes with increasing precision and dynamics. Among them, hexahedral meshes are commonly handled in biomedical engineering [2], computational materials science [3], and in geosciences. They are for instance used by geologists to study flow simulations for reservoir modelling [4], and benefit from an increasing interest for geologic model building [5, 6]. Huge progresses in data acquisition produce increasingly more accurate digitization of physical terrains. The tremendous quantity of data thus generated prominently impacts computational resources and performances: memory size required for their storage and visualization, but also their transmission and transfer, and ultimately their processing. Consequently, it greatly affects the overall simulation interactivity. This trend affects the oil and gas sector at large [7].

We propose HexaShrink, an efficient multiscale representation dedicated to hexahedral meshes with attributes and discontinuities. HexaShrink combines four wavelet-like decompositions to adapt to the heterogeneous nature of geoscience meshes. Geometrical, continuous, and categorical properties are consistently downsampled (upscaled in geoscience terms [8]) in an exactly reversible manner. In addition to regular structures, HexaShrink also strives

This work was partly presented in [1].

✉ Laurent Duval
Laurent.Duval@ifpen.fr

Extended author information available on the last page of the article.

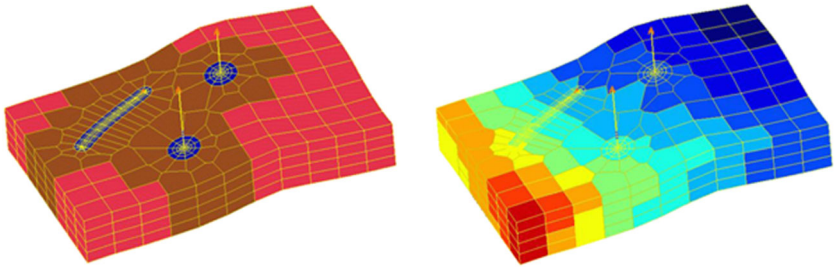


Fig. 1 Example of a VM used in geosciences (left); same mesh with the associated porosity property (right). Note that this mesh is hybrid and unstructured, with both hexahedral and tetrahedral elements

to manage mesh externalities like boundaries and borders tagged as inactive cells for simulation purposes. It produces a hierarchy of meshes at dyadic resolutions maintaining geometrical coherency over scales, consistent with geomodeler/simulator upscaling operations [8]. It finally lends itself to efficient lossless storage, in combination with state-of-the-art compression algorithms.

The paper is structured as follows: Section 2 presents the specificities of structured hexahedral meshes and the discontinuities they may contain. Section 3 reviews prior methods for volume mesh representation or compression [9]. We introduce the HexaShrink structured mesh representation in Section 4. We detail the four main multiscale wavelet-like schemes that entail an exactly invertible hierarchy of downsampled meshes, consistently with respect to geometry and properties. A special care is taken on the accurate representation of faults and the management of mesh borders. Section 5 presents visual results and evaluates the quality of the HexaShrink with respect to categorical property coherency across scales, and dyadic upscaling by a geomodeler. An exhaustive evaluation obtained from combining HexaShrink with different lossless compression algorithms, at different resolution levels shows the interest of the proposed representation in terms of lossless compression for storage. Finally, Section 6 summarizes our contributions and proposes future works.

2 Volume meshes: a primer

2.1 Generic definitions

Volumetric or volume meshes (VMs) discretize the interior structure of $3D$ objects. They partition their inner space with a set of three-dimensional elements named cells (or zones). While pyramid and triangular prism partitions exist, most of the existing VMs are composed of tetrahedral (4 faces) or *hexahedral* (6 faces) elements. They are called tets or hexes (sometimes bricks), respectively. A VM composed of different kinds of cells, tetrahedra and hexahedra for instance, is termed hybrid. A VM is described by the location of vertices in $3D$ space (*geometry*) and the incidence information between cells, edges, and vertices (*connectivity*). In function of the application domain, VMs also contain *physical properties* associated to vertices, edges, or cells. In geosciences, properties can be scalar (like a single porosity value, Fig. 1) or vectorial (a vector of compositional proportions in different rocks within a cell…). We also distinguish *categorical* or *nominal* variables (symbolic and discrete values describing the composition of rocks: sandstone (0), limestone (1), shale (2)…) from *continuous properties* (saturation, porosity, permeability, or a temperature taking values in a given range $[-T_\flat, T_\sharp]$).

A non-degenerate hex has 6 *faces* named quads, 12 *edges*, and 8 *vertices*. Depending on incidence information between cells, edges, and vertices, hexahedral meshes are either unstructured or structured. The degree of an edge is the number of adjacent faces. An hexahedral mesh is *unstructured* if cells are placed irregularly in the volume domain, i.e., if degrees are not the same for all edges of the same nature. Unstructured meshes have an important memory footprint, as all the connectivity information must be described explicitly. However, they are well-suited to model complex volumes, Computer-aided design (CAD) models for instance, as shown in Fig. 2.

An hexahedral mesh is *structured* if cells are regularly organized in the volume domain, i.e., if the degree is equal to four for *interior* edges (inside the volume), and equal to two for *border* edges (on a border of the volume). In that case, the set of hexahedral cells is topologically aligned on a $3D$ Cartesian grid (see Fig. 3). Each vertex of the mesh can be associated to a node of the grid. Hence, each cell can be indexed by only one triplet (i, j, k), and the connectivity information becomes implicit: only the position of the vertices is needed to model the mesh.

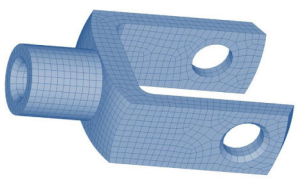


Fig. 2 CAD model defined by an unstructured VM

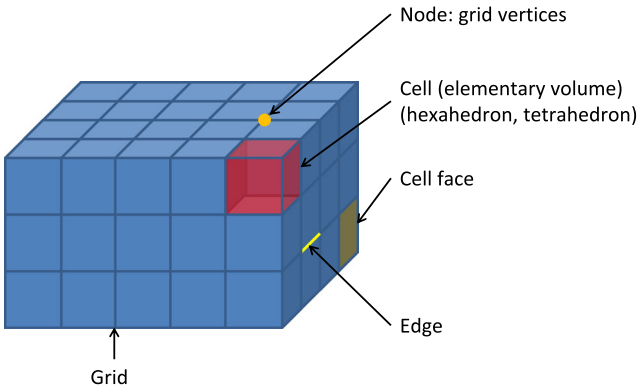


Fig. 3 Structured hexahedral mesh composed of $(5 \times 4 \times 3)$ cells

2.2 Hexahedral meshes & geometrical discontinuities

Hexahedral meshes in geosciences are generally structured, and thus based on a Cartesian grid. But these meshes may contain *geometrical discontinuities*. They correspond, for instance, to geological faults. It induces a vertex disparity in space at the same node. The association of one node of the Cartesian grid with 8 vertices (one for each adjacent cell) handles this specificity. Figure 5a provides an illustration of a fault-free volume. On Fig. 5b, we see that this structure allows to describe for instance a vertical fault (by positioning vertices differently about the node), while preserving the Cartesian grid.

The most popular data structure for structured hexahedral meshes with geometrical discontinuities is the *Corner Point Grid* [10, 11] tessellation of an Euclidean $3D$ volume. This structure is often termed *pillar grid*. It is based on a set of vertical or inclined pillars running from the top to the bottom of the geological model. A cell is defined by its 8 adjacent vertices (2 on each adjacent pillars, see Fig. 4), and the vertices of the adjacent cells are described independently, in order to model faults and gaps. Across the associated Cartesian grid, each cell can be indexed by a triplet (i, j, k).

This pillar grid also allows to model geological collapses (or erosion surfaces), by using *degenerate* cells, i.e., cells with (at least) two vertices on one pillar located at the same position (see Fig. 6 for different degenerate configurations).

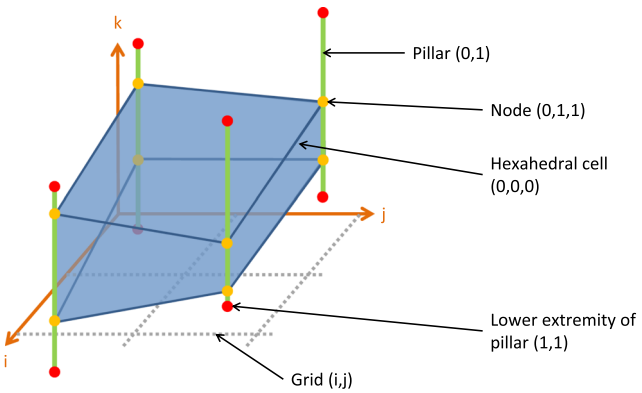


Fig. 4 An hexahedron, according to the *pillar grid* structure

3 Mesh compression: an overview

We deal in this section mostly with the ontological description of volume meshes, leaving aside the specificities related to actual data format.

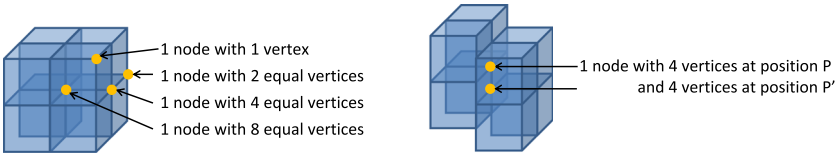
3.1 Basic techniques for mesh encoding

The most straightforward technique to encode a VM is to use an indexed data structure: the list of all the vertex coordinates (three floating-point values, which amounts to 96 bits par vertex), followed by their connectivity. The connectivity is defined cell after cell, each cell being defined by the set of indexes of the adjacent vertices (8 integers per hex). Physical property encoding depends on their nature: categorical/continuous, dimension, associated to cells or vertices. They thus only provide estimates of an actual compression performance.

To reduce the memory footprint or make the transmission of VMs faster, well-known techniques exist. The simplest tool for the geometry is the *quantization* of vertex coordinates. It consists of constraining the vertex coordinates to a discrete and finite set of values. Hence, it becomes possible to encode each coordinate with an integer index, instead of a 32-bit floating-point value. It is common to quantize the coordinates with 12 or 16 bits, reducing the geometry information by a compression factor of 2.6 or 2 respectively. Quantization inevitably introduces an irreversible loss in accuracy. Visualization typically tolerates precision loss (as long as visual *distortion* remains negligible), unlike some numerical simulations requiring more precise computations.

Prediction (as well as related interpolation methods) further improves the geometry compactness. Predictive coding resorts to estimating the position of a vertex from already encoded neighbor vertices. *Prediction errors* (differences between predicted and actual positions) are generally smaller in amplitude and sparser, which makes their entropy coding (which codes differently frequently occurring patterns) efficient [12, p. 63 sq.].

Regarding connectivity, when meshes are unstructured, the most frequent technique performs a traversal of mesh elements and describes the incidence configurations with a reduced list of symbols. These symbols are then entropy coded. When meshes are structured, the connectivity is implicit, reducing its cost to zero. For such meshes, the



(a) Free-fault area. (b) Area with a vertical fault.

Fig. 5 A fault-free and a fault area. **a** Free-fault area. **b** Area with a vertical fault

only additional information to encode are geometrical discontinuities describing faults and gaps.

3.2 Volume mesh compression: prior works

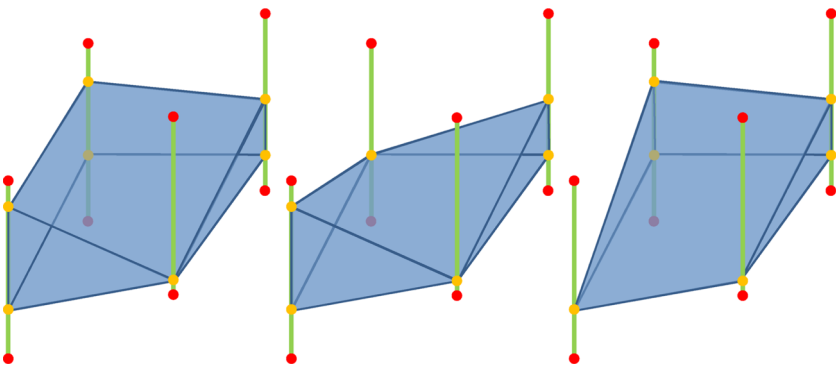
The basic tools previously presented can be implemented on the ontological structure of meshes and improved in many different ways. Their combination, with the assistance of advanced compression techniques, permits more efficient tetrahedral or hexahedral mesh coding. Previously proposed algorithms are presented below, classified into two categories: *single-rate* and *progressive/multiresolution*.

3.2.1 Single-rate mesh compression

They lead to a compact mesh representation, most of the time driven by efficient connectivity encoding. The first method for tetrahedral meshes, Grow & Fold, was presented by Szymczak and Rossignac [13] at the end of the nineties. It is an extension of *EdgeBreaker* [14] developed for triangle meshes. The method consists in building a tetrahedral spanning tree from a root tetrahedron. The traversal is arbitrary among the three neighboring tets (Section 2.1) of the cell currently processed, and 3 bits are needed to encode each cell. The resulting spanning tree does not retain the same topology as the original mesh, because some vertices are replicated during the traversal. "Fold" and "glue" techniques are thus needed during encoding to restore the original mesh from the tetrahedron tree. The additional cost is 4 bits, leading to a total cost of 7 bits per tetrahedron.

The *cut-border* initiated in [15] was adapted to tetrahedral meshes [16]. It denotes the frontier between tetrahedra already encoded and those to encode. At each iteration, either a triangle or an adjacent tetrahedron is added to the cut-border. In this case, if the added vertex is not already in the cut-border, this latter is included by a connect operation and is given a local index. As the indexing is done locally, the integers to encode are very small, leading to a compact connectivity representation. In addition, two methods are proposed to encode geometry and associated properties, based on prediction and entropy coding. This method yields good bit rates (2.4 bits per tetrahedron for connectivity) for usual meshes, handles non-manifold borders, but worst-cases severely impact bitrates and runtimes (which tend to be quadratic).

Isenburg and Alliez [17] are the first to deal with hexahedral VMs. The connectivity is encoded as a sequence of edge degrees—in a way similar to [18] for triangular meshes—*via* a region-growing process of a convex hull called *hull surface*. It relies on the assumption that hexahedral meshes are often highly regular, which implies that the majority of vertices are shared by 8 cells. It involves an almost constant edge degree all over the mesh, which significantly decreases the entropy of the connectivity information. A context-based arithmetic coder [19] is then used to encode the connectivity at very low bit rates, between 0.18 and 1.55 bits per hexahedron. Regarding geometry, a user-defined quantization first restricts the number of bits for coordinates, and then a predictive scheme based on the parallelogram rule encodes the position of vertices added during the region-growing process.



(a) Single. (b) Two opposed. (c) Two adjacent.

Fig. 6 Degenerate hexahedral cells due to a single collapsed pillar (**a**) or two different collapsed pillar locations (**b**), (**c**)

Krivograd et al. [20] propose a variant to [17] that encodes the vertex degrees—number of non-compressed hexahedra around a given vertex—instead of the edge degrees. On the one hand, this variant achieves better compression performances than [17] for dense meshes. On the other hand, it only deals with manifold meshes, and the algorithm is complex as interior cells are encoded after border cells (it involves many specific cases to process when encoding the connectivity).

Lindstrom and Isenburg proposed an original algorithm for unstructured meshes called Hexzip [21]. This algorithm is considered as fully lossless, because the initial indexing of vertices and hexahedra is preserved. For this purpose, connectivity is encoded directly in its indexed structure, by predicting the eight indices of an hexahedron from preceding ones. This technique is suitable because hexahedral meshes generally have regular strides between indices of subsequent hexahedra. A hash-table is then used to transform the index structure into a very redundant and byte-aligned list of symbols, that can be compressed efficiently with gzip (discussed in Section 5.4). Concerning the geometry, spectral prediction [22] is used. This algorithm is faster and less memory intensive than [17] as the connectivity is not modified. It handles non-manifold meshes and degenerate elements. On the other hand, bitrates are higher because of the lossless constraints. Unlike methods presented above, Chen et al. [23] focus on geometry compression for tetrahedral meshes. The authors proposed a *flipping* approach based on an extension of the *parallelogram rule* (initially proposed for triangle meshes [18]) to tetrahedra. It consists in predicting the position of an outer vertex of two face-adjacent tetrahedra, with respect to the other vertices. To globally optimize the geometry compression, a minimum spanning tree minimizing the global prediction error for the whole mesh is computed. This method is more efficient than prior flipping approaches whose traversal does not depend on the geometry, but solely on the connectivity.

Streaming compression is a subcategory of single rate compression, dedicated to huge data that cannot fit entirely in the core memory. A particular attention to I/O efficiency is thus required, to enable the encoding of huge meshes with a small memory footprint. Isenburg and coworkers are the first to propose streaming compression for VMs (extended from his method for triangular meshes [24]): for tetrahedral meshes [25], and then for hexahedral meshes [26]. In the latter, for instance, the compressor does not require the knowledge of the full list of vertices and cells before encoding. The compressor starts encoding the mesh as soon as the first hexahedron and its eight adjacent vertices have been read. For a given hexahedron: (i) its face adjacency is first encoded in function of its configuration with hexahedra already processed; (ii) positions of vertices that are referenced for the first time are predicted (spectral prediction from adjacent cells); (iii) prediction errors are encoded; (iv) data structures relative to vertices, becoming useless (because their incidence has been entirely described) are finally removed from memory. Compared to other single rate techniques, streaming tends to achieve similar compression performances for geometry, but poorer performances for connectivity.

3.2.2 Progressive/multiresolution mesh compression

Progressive algorithms (also called scalable or *multiresolution*, see Section 4.1 for details) enable the original meshes to be represented and compressed at successive LODs (levels of details). The main advantage is that it is not necessary to decompress a mesh entirely before vizualising it. A coarse approximation of the mesh (also known as its lowest resolution) is first decompressed and displayed. Then this coarse mesh is updated with the successive LOD (termed higher resolutions) that are decompressed progressively. While they cannot achieve yet compression performance of single-rate algorithms, progressive algorithms are popular because they enable LOD, and also adaptive transmission and displaying, in function of user constraints (network, bandwidth, screen resolution...).

Pajarola et al. [27] are the first to propose in 1999 a progressive algorithm dedicated to VM compression. This work is inspired by a simplification technique for tetrahedral meshes [28]. It simplifies a given tetrahedral mesh progressively, by using successive *edge collapses* [29]. Each time an edge is collapsed, its adjacent cells are removed, and all the information required to reverse this operation is stored: index of the vertex to split, and the set of incident faces to "cut." Thus, during decompression, the LODs can be also recovered iteratively, by using the stored data describing *vertex splits*. During coding, an edge is selected such as its collapse leads to the minimal error, with respect to specific cost functions. This algorithm gives a bitrate inferior to 6 bits per tetrahedron (for connectivity only).

In 2003, Danovaro et al. [30] propose two progressive representations based on a decomposition of a field domain into tetrahedral cells. The first is based on *vertex splits*, as the previous method, the second is based on *tetrahedron bisections*. This operation consists in subdividing a tetrahedron into two tetrahedra by adding a vertex in the middle of its longest edge. Unlike with vertex splits, the representation based on *tetrahedron bisections* is obtained by following a coarse-to-fine approach, i.e., by applying successive bisections to an initial coarse mesh. Also, this representation only needs to encode the difference vectors between the vertices added by bisections and theirs real positions. This representation is thus more compact, as

the mesh topology does not need to be encoded, but it only deals with structured meshes.

VMs multiresolution decomposition based on wavelets [31] was proposed by Boscardín et al. [32] for tetrahedral meshes. It is based on the tetrahedron subdivision scheme [33] that transforms a tetrahedron into 8 subtetrahedra, by introducing 6 new vertices on each edge. After analysis, the input mesh is replaced by a base tetrahedral mesh, and several sets of detail wavelet coefficients. Although coefficients corresponding to differences between two resolutions could be encoded for mesh synthesis, this work does not provide an actual compression scheme.

In [34], Chizat proposed a prototype for a multiresolution decomposition of geoscientific hexahedral meshes with the pillar grid structure (Section 2.2). His main contribution resides in a multiresolution analysis (MRA) that partially manages geometrical discontinuities representing the faults. It can be achieved by using a morphological wavelet transform (Section 4.2.2). This non-separable transform enables the preservation of some fault shapes at different resolutions, as depicted in Fig. 7.

The latter work is a seed for the upcoming description of HexaShrink.

4 Global HexaShrink algorithmic workflow

4.1 Multiresolution analysis: background

MRA or multiscale *approximation* can be interpreted as a decomposition of data at different resolutions, LOD or scales, through a recursive *analysis* process. It is called exact, reversible, or invertible when a *synthesis* scheme can retrieve the original data. Inter-scale relationships [35, 36] often yield *sparsification* or increased *compressibility* on sufficiently regular datasets. In discrete domains, each *analysis* stage transforms a set of values (continuous or categorical, in one or several dimensions), denoted by S^0. The resulting representation consists in one subset S^{-1} of approximation coefficients at a lower resolution identified by a negative index S^{-1} that approximates the original signal, *plus* one subset of details D^{-1} or a combination thereof. The latter represents information missing in the approximation S^{-1}. Depending on the MRA scheme, the lower *resolution* S^{-1} may represent a coarsening or "low frequencies" of the original samples, or an upscaling in geosciences (cf. Section 4). The subset D^{-1} represents refinements, fast variations or "high-frequency" details removed from S^0. We consider here exact systems, allowing the perfect recovery of S^0 from a combination of subsets S^{-1} and D^{-1}. Hence, a similar analysis stage can be applied iteratively, and perfectly again, to the lower resolution S^{-1}, in a so-called pyramid scheme. Thus, with the non-positive extremum decomposition level L, and indices $0 \geq l \geq L$, after an $|L|$-level multiresolution decomposition, the input set S^0 is now decomposed and represented by the subset S^L—a (very) coarse approximation of S—and $|L|$ subsets of details $D^L, \ldots, D^l, \ldots, D^{-1}$, representing information missing between each two consecutive approximations.

We consider in the following four different MRA flavors, all called wavelets for simplicity. They stem from iterated, (rounded) linear or non-linear combinations of coefficients, as well as separable (applied separately in $1D$ on each direction) or non-separable ones. Without going into technicalities here (cf. Section 4.2.3), computations are performed using the *lifting scheme*. It suffices to mention that lifting uses complementary interleaved grids of values, often indexed with odd and even indices. Values on one grid are usually predicted (approximations) and updated (details) from the others. The main interests reside in reduced computational load, in-place computations and the possibility to maintain exact integer precision, using for instance only dyadic-rational coefficients (written as $m/2^n$, $(m, n) \in \mathbb{Z} \times \mathbb{N}$) and rounding. We refer to [31, sections 2.3, 3.2, and 4.3] for a concise account on both non-separable and non-linear wavelet MRAs, and to [37–40] for more comprehensive visions of wavelets and their

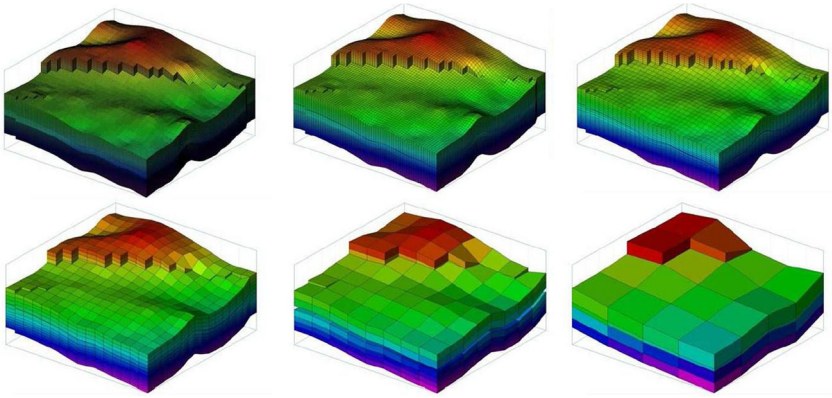


Fig. 7 Dyadic non-separable multiresolution rendering on a simple geologic mesh [34]

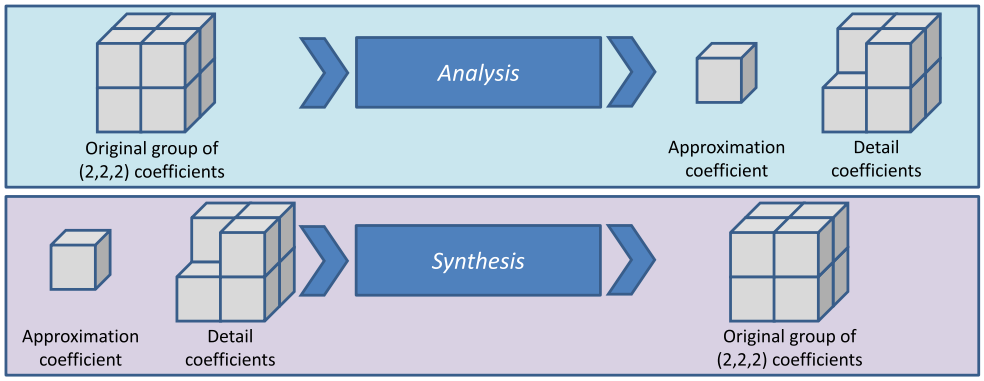


Fig. 8 Analysis and synthesis stages for VMs

lifting implementations. A recent use in geological model upscaling is given in [41].

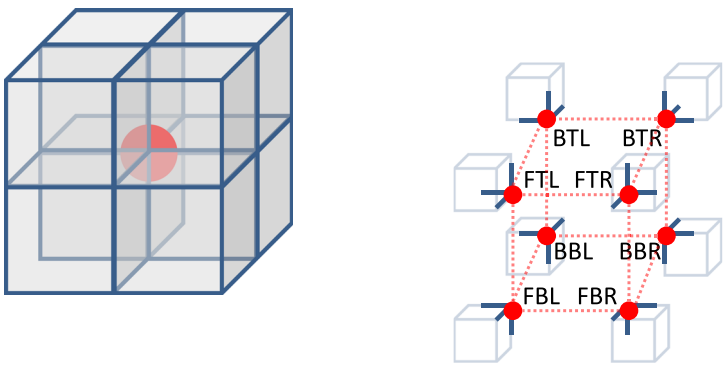
More simply put, for our hexahedral VMs, the dyadic analysis stage transforms each cell block $\mathcal{C}^l$ of values around $2^3 = 8$ contiguous cells (possibly borrowing values from a limited cell neighborhood) at resolution l. They are turned into one *approximating cell* (lower resolution (S^{l-1}), and a subset of $2^3 - 1 = 7$ detail cells D^{l-1}, as depicted in Fig. 8 along with the reverse synthesis stage. Hence, if a VM at resolution l is composed of $\left(\mathcal{C}_i^l \times \mathcal{C}_j^l \times \mathcal{C}_k^l\right)$ cells, $\mathcal{C}_i^l$ being the number of cells in direction i, the VM of lower resolution will be of dimension $\left\lceil \frac{\mathcal{C}_i^l}{2} \right\rceil \times \left\lceil \frac{\mathcal{C}_j^l}{2} \right\rceil \times \left\lceil \frac{\mathcal{C}_k^l}{2} \right\rceil$, to take into account non-power-of-two sized grids. As several digital attributes are associated with each cell (geometry, continuous or categorical properties), different types of MRA are performed separately on the different variables defining these properties, as explained in the following sections.

4.2 Multiresolution scheme for geometry

Standard linear MRA schemes rely on smoothing or averaging and difference filters for approximations and details, respectively. To preserve coherency of representation of geometrical discontinuities—whatever the resolution—a special care is taken to avoid excessive smoothing, while at the same time allowing the reverse synthesis. As the pillar grid format is used (see Section 2.2), vertices are inevitably positioned along pillars. So, our multiresolution scheme for geometry data only focuses on:

- the z coordinates of the 8 vertices associated to each node. According to the naming convention presented in Fig. 9, those 8 vertices can be differentiated according to their relative positions [back (B)/front (F), bottom (B)/top (T), left (L)/right (R)];
- the x and y coordinates of the nodes describing the low (bottom) and high (top) extremities of all the pillars (the x and y coordinates of intermediary nodes being implicit). The nodes are called hereinafter the *floor* and *ceil* nodes, respectively.

Actual $3D$ meshes can exhibit very irregular boundaries. Hence, a Boolean field called ACTNUM may be associated to each cell to inactivate its display (and its influence during simulations as well). It enables the description of either mesh boundaries (Fig. 10), or caves/overhangs. Resultantly, this ACTNUM field must be carefully considered during



(a) A node and its 8 surrounding cells. (b) Splitting view of the node into its 8 vertices.

Fig. 9 Vertex naming with the back (B)/front (F), bottom (B)/top (T), left (L)/right (R) convention. **a** A node and its 8 surrounding cells. **b** Splitting view of the node into its 8 vertices

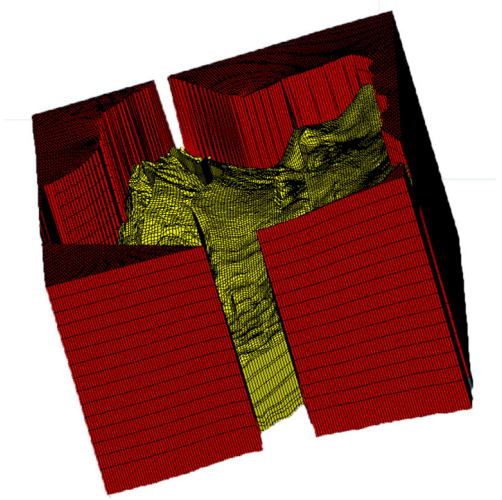


Fig. 10 Mesh#5 (in yellow) has inactive cells (in red) to describe its boundaries using the ACTNUM field

the multiresolution analysis of the geometry data, to avoid artifacts at lower resolutions on frontiers between active and inactive cells (see Section 4.2.4 and Fig. 16).

By construction, most geological VMs have no horizontal fault, as there is no vertical gap between any two adjacent layers of cells. For every node, each of the four top vertices has the same z coordinate as its counterpart bottom vertex. Therefore, from now on, our geometry multiscale representation method only deals with the z coordinates of the bottom vertices BBL, BBR, FBL, and FBR of each node.

An instance of the decomposition shown in Fig. 8 can be implemented with the proposed two-step technique depicted by arrows in Fig. 11:

- A non-linear and non-separable $2D$ **morphological wavelet transform** applied on the nodes, in order to detect the faults in the input VM, and then to preserve their coherency in the lower resolutions. This step relies on a **fault segmentation** within the input VM obtained by studying all possible fault configurations for the top view of the VM (see Fig. 12);
- A non-linear $1D$ **wavelet transform** applied on the output of the above first step to analyze **the z coordinates of the vertices** along each pillar. The same $1D$ wavelet transform is also applied on the sets of x and y coordinates of the *floor* and *ceil* nodes, to complete the "horizontal" decomposition.

4.2.1 Fault segmentation

This stage detects the faults in the original mesh, in order to preserve them during the morphological wavelet analysis. For each node, a dozen of fault configurations, depending on BBL, BBR, FBL, and FBR, is possible: fault-free (1), straight (2), corner (4), T-oriented (4) or cross (1), as illustrated in Fig. 12.

Each configuration depends on the four orientations of the cardinal axes (north, south, east, and west), which are either active or inactive. For instance, the T-north configuration has its south axis inactive, while the three remaining ones are active. Assuming that a fault configuration is z-invariant, meaning that the nodes belonging to the same pillar present the same fault configuration, a single $2D$ configuration map is sufficient to represent the fault configuration of the whole mesh, as illustrated by Fig. 13.

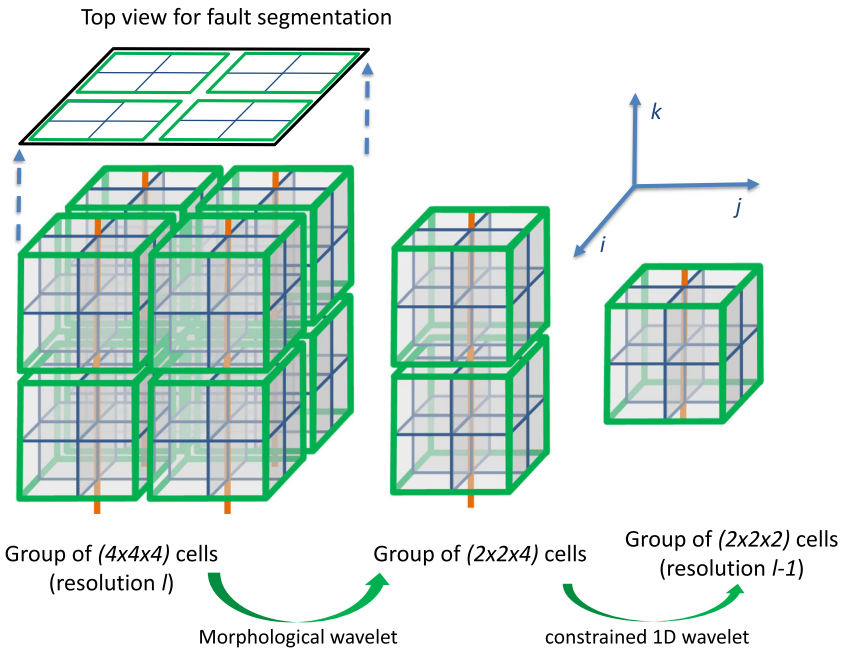


Fig. 11 HexaShrink multiresolution scheme for geometry: (left) input grid and its top view; (middle) output from the non-separable, non-linear $2D$ morphological wavelet based on a fault segmentation (based on the top view); (right) non-linear $1D$ wavelet transform along pillars (orange lines)

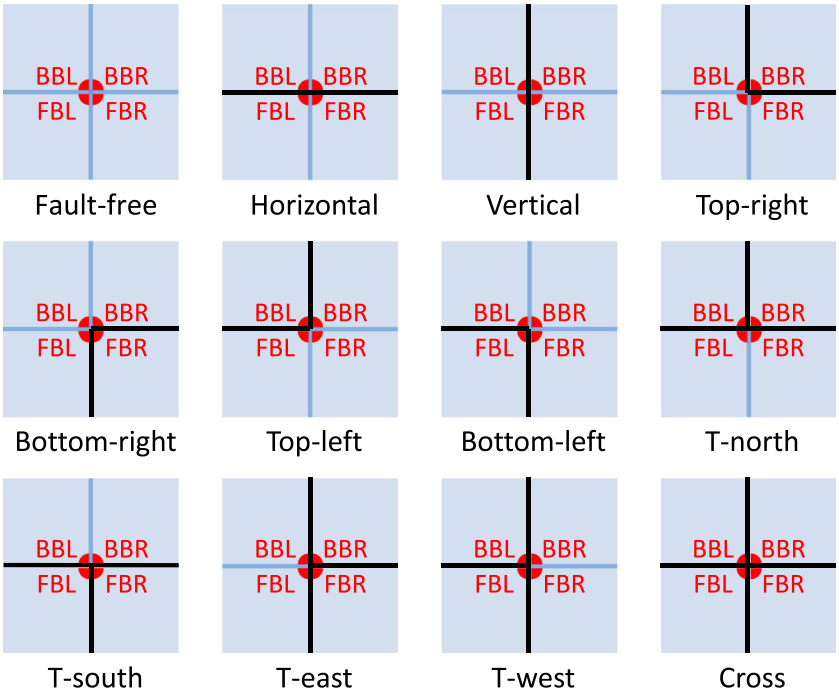


Fig. 12 The 12 possible fault configurations (in black lines) at a given node

4.2.2 Horizontal 2*D* morphological wavelet transform

The fault segmentation guides the multiresolution analysis to preserve faults, as much as possible, all over the decomposition process. The fault configuration of 4 associated nodes at resolution l is used to predict the extension of the downsampled fault structure at resolution $l-1$.

This horizontal prediction is based on the logical function OR ($\vee$), computed on each side of each group of 4 nodes. For instance, a resulting fault node configuration contains a west axis if the fault configurations of the 2 left nodes contain at least 1 west axis, as illustrated in Fig. 14. By repeating the procedure for the north, south and east axes of each resulting node, fault node configurations at lower resolutions are fully predicted. This non-linear and peculiar choice is meant to maintain a directional flavor of orientated faults for flows; other choices could be devised, depending on physical rules and geological intuitions.

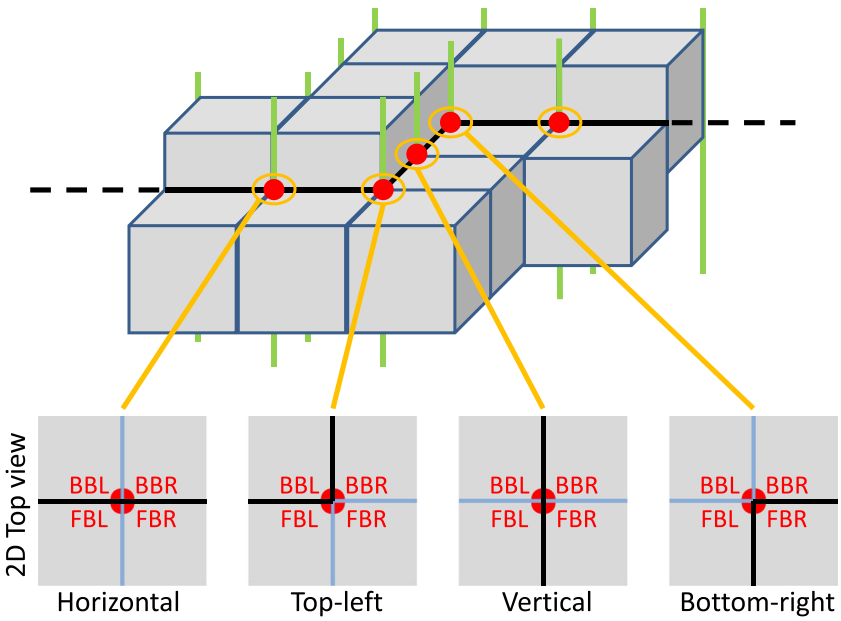


Fig. 13 Fault segmentation within the original mesh

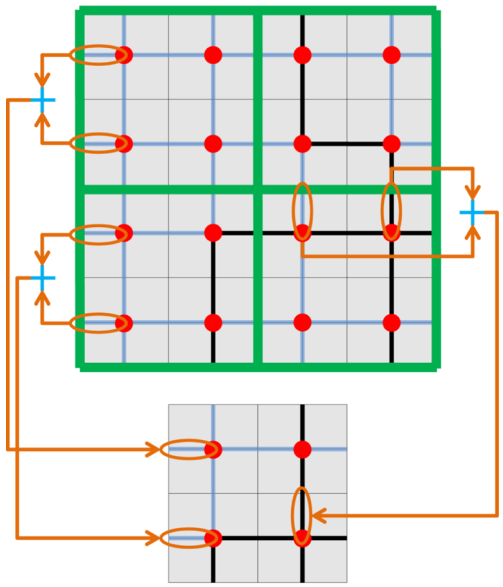


Fig. 14 Prediction of a fault node at resolution $l-1$ from the four parents' configuration at resolution l, orange ovals denoting $\vee$ operands

Finally, from this prediction, the node whose configuration minimizes its distance with the predicted one, corresponds to the aforementioned approximation coefficient, which will be part of the novel Z matrix at lower resolution $l-1$. The same procedure can be applied recursively until the wanted resolution.

4.2.3 Rounded linear 1*D* wavelet transform

This $1D$ wavelet transform is applied on the output of the above horizontal $2D$ morphological wavelet, to analyze the z coordinates of the 4 sets of vertices BDR, FDR, BDL, and FDL separately, along each selected pillar. Hence, HexaShrink here decomposes at each scale z^l into a subsampled pillar coordinate z^{l-1} and its associated detail d^{l-1}. By geomodel construction, coordinate behavior along the pillars is expected to be relatively smooth. This entails the use of a modified, longer spline wavelet. The latter can be termed LeGall [42], or CDF 5/3 (after Cohen, Daubechies, and Feauveau [43]), or biorthogonal 2.2 from its vanishing moments.

The lifting analysis operations *Prediction* and *Update* are depicted by Fig. 15. To retrieve respectively the sets of details d^l and the z^l coordinates at resolution $l-1$ from scale l, the following equations are used ($\forall n \in \mathbb{N}$):

$$d^{l-1}[n] = z^l[2n+1] - \left\lfloor \frac{z^l[2n] + z^l[2n+2]}{2} \right\rfloor, \quad (1)$$

$$z^{l-1}[n] = z^l[2n+0] + \left\lfloor \frac{d^{l-1}[n-1] + d^{l-1}[n]}{4} \right\rfloor, \quad (2)$$

where both dyadic integers and rounding are evident (see Section 4.1). With rounding, lifting schemes can thus manage integer-to-integer transformations [44]. For synthesis, to reconstruct resolution l from resolution $l-1$, we only have to reverse the order and the sign of the equations:

$$z^l[2n] = z^{l-1}[n] - \left\lfloor \frac{d^{l-1}[n-1] + d^{l-1}[n]}{4} \right\rfloor, \quad (3)$$

$$z^l[2n+1] = d^{l-1}[n] + \left\lfloor \frac{z^l[2n] + z^l[2n+2]}{2} \right\rfloor. \quad (4)$$

4.2.4 Managing externalities: borders and boundaries

A pertinent multiresolution on complex meshes requires to cope with externalities that may hamper their handling: floor and ceil borders and outer boundaries (Fig. 10). First, to keep borders unchanged from the original mesh, throughout all resolutions, the following constraints must be met:

$$z^{l-1}[0] = z^l[0], \quad (5)$$

$$z^{l-1}[n_k^{l-1} - 1] = z^l[n_k^l - 1]. \quad (6)$$

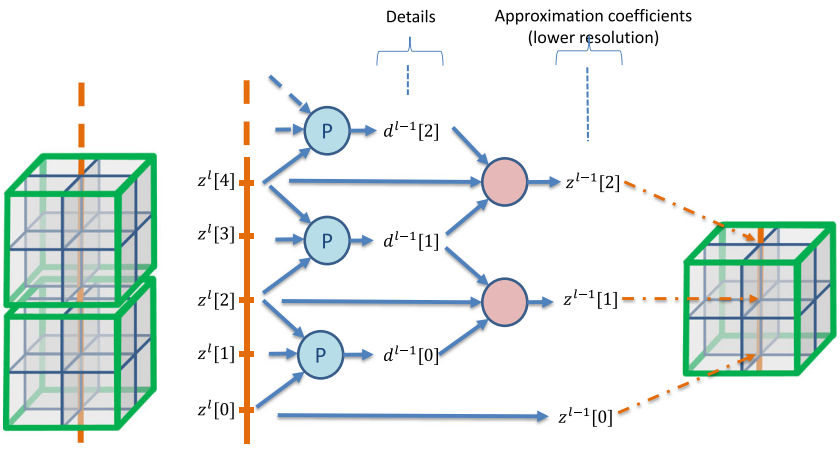


Fig. 15 Principle of the lifting scheme (prediction and update) for the rounded linear $1D$ wavelet from Eqs. 1–2, to analyze z coordinates of vertices BDR, FDR, BDL, and FDL along each pillar, as well as x and y coordinates of the *floor* and *ceil* nodes

Both constraints can be fulfilled if one satisfies the following conditions:

– Floor border condition to meet (5):

$$d^{l-1}[-1] = -d^{l-1}[0]\,, \tag{7}$$

– Ceil border condition to meet (6):

$$d^{l-1}[n_k^{l-1} - 1] = -\,d^{l-1}[n_k^{l-1} - 2]\,, \quad (n_k^l \text{ odd}) \tag{8}$$

$$d^{l-1}[n_k^{l-1} - 1] = -\,d^{l-1}[n_k^{l-1} - 2] \quad (n_k^l \text{ even}) + 4z^l[n_k^l - 1] - 4z^l[n_k^l - 2]\,. \tag{9}$$

To complete the MRA of the geometry, the same rounded $1D$ wavelet as in Section 4.2.3 is finally applied to the sets of *floor* and *ceil* nodes of the initial VM, to get the x and y coordinates of the extremities of the remaining pillars at the lower resolution.

Second, the ACTNUM field should also be considered to lessen mesh boundary artifacts. Indeed, severe disturbances may appear at lower resolutions if not wisely processed during analysis, as shown in Fig. 16.

A cell is deemed active if and only if its 8 adjacent vertices are active at the resolution l. During our study, we found that one vertex at resolution $l-1$ could be considered active if and only if its parent vertices selected by the morphological wavelet at resolution l (Section 4.2.2) are active. So, a cell at resolution $l-1$ is considered active if and only if its 8×2 corresponding parent vertices are active at resolution l.

4.3 Multiresolution scheme for continuous properties

Once the geometry is coded, one can focus on associated continuous properties. For scalar ones, a value $p_i \in \mathbb{R}$ is associated to each cell i in the mesh. Consistently with the handling of cell blocks $\mathcal{C}$ of $2 \times 2 \times 2$ cells throughout scales, we use an adaptation of the well-known *Haar* wavelet. The resolution $l-1$ is a scaled average of cells at resolution l. The approximation coefficient p^{l-1} is thus the average value of the related eight property coefficients $\{p_1^l, p_2^l, \ldots, p_8^l\}$. The seven details required for synthesis are differences with respect to the approximation coefficient:

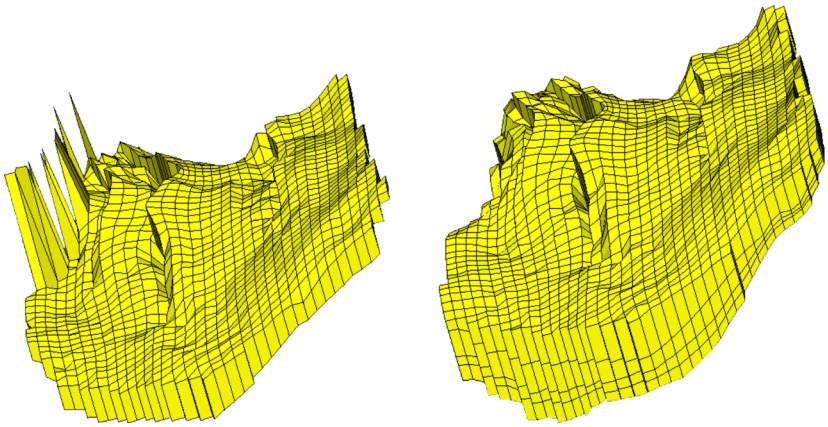
$$p^{l-1} = \frac{1}{8}\sum_{n=1}^{8} p_n^l\,; \quad d_n^{l-1} = p_n^l - p^{l-1}\,,\ \forall n \neq 1\,.$$

To deal with real-valued (floating-point) properties, and avoid accuracy imprecision due to the divide operator, we introduce the following modifications. First, reals are mapped into integers up to a user-defined precision, here with a 10^6 factor. Second, we disable the division by using a sum. The analysis system thus becomes:

$$p^{l-1} = \sum_{n=1}^{8} p_n^l\,; \quad d_n^{l-1} = 8p_n^l - p^{l-1}\,,\ \forall n \neq 1\,,$$

and the synthesis system turns into:

$$p_n^l = \frac{1}{8}(d_n^{l-1} + p^{l-1})\,,\ \forall n \neq 1\,; \quad p_1^l = p^{l-1} - \sum_{n=2}^{8} p_n^l\,.$$



(a) Boundary artifacts without proper ACTNUM management.

(b) Lower mesh resolution with efficient ACTNUM Boolean values care-taking.

Fig. 16 Inadequate ACTNUM fields management during analysis may lead to severe boundary artifacts (left) that can be dealt with (right) as *examplified* with mesh#5 from Fig. 10. **a** Boundary artifacts without proper ACTNUM management. **b** Lower mesh resolution with efficient ACTNUM Boolean values care-taking

Approximation and coefficients are stored as is. To recover the accurately scaled values, the division operator should however be applied as a simple linear post-processing.

4.4 Multiresolution scheme for categorical properties

We finally complete the global mesh multiresolution decomposition with an original categorical-valued scheme called *modelet* [45]. We assume that a mesh cell category belongs to a set of classes $\Omega_0 = \{\omega_1, \omega_2, ..., \omega_W\}$, taking discrete values. The cell block $\mathcal{C}^l$ $\{p_1^l, p_2^l, \ldots, p_8^l\}$ thus contains, at resolution l, integers indexing categories from Ω_l. They take values in a subset of Ω. The multiresolution scheme is expected to produce, at lower resolutions, discrete values in embedded subsets: $\Omega_0 \supset \Omega_{-1} \supset \cdots \supset \Omega_l \supset \cdots$. In other words, a cell category can only belong to an existing category at an upper resolution. We choose here the modal value (mode), i.e., the most frequently represented in $\mathcal{C}^l$. If $|\omega_w|$ denotes the cardinal of this class, then $\sum_{w=1}^{W} |\omega_w^l| = |\mathcal{C}^l| = 8$. We choose for the modelet:

$$p^{l-1} = \arg\max\{|\omega_w^l|, \omega_w^l \in \Omega_l\}.$$

It may happen that the above definition does not yield a unique maximum. If two or more categories dominate a cell block, a generic approach consists in taking into account its first block cell neighborhood (the surrounding 26 cells, except at mesh borders and boundaries). We affect the dominant value in the first neighborhood to p^{l-1}. In case of a draw again, the second-order surrounding can be used, iteratively. In practice for the presented version of HexaShrink, we limit to the first-order neighborhood, and choose the lowest indexed category when the maximum is not unique. Equipped with this unique lower resolution representative value, we proceed similarly to Section 4.3 for details, by using differences between original categories and the mode. As classes are often indexed by positive integers, a slight motivation allows to get only non-negative indices. By avoiding negative values, one expects a decrease in data entropy of around 5%, which benefits to compression.

We thus change the sign of a detail coefficient if and only if it generates a value out of the range of $\{\omega_1, \omega_2, \ldots, \omega_W\}$, and then control this condition during reconstruction. So, all details $\{d_n^{l-1}\}$ for a cell block $\mathcal{C}$ are determined by:

$$d_n^{l-1} = (-1)^{(p_n^l - p^{l-1} < 0) \wedge ((2p^{l-1} - p_n^l) \notin \Omega)} \times (p_n^l - p^{l-1}).$$

During synthesis, coefficients $\{p_i^l\}$ are obtained thanks to the closed-form equation:

$$p_n^l = p^{l-1} + (-1)^{((p^{l-1} + d_n^{l-1}) \notin \Omega)} \times d_n^{l-1}.$$

5 Evaluation methodology, comparative results, and discussion

5.1 Evaluation methodology

Meshes, hexahedral ones in particular, are complex composite objects. The ontological description of their geometry is subject to different options, "Block Centered" or "Corner Point" grids for instance. Their generation, cell size, and resolution for practical applications may have undergone more or less complex processing. Mesh complexity can range from simply layered, homogeneous modes to massively faulted environment with highly varying properties. Encoded numerical values, albethey cell coordinates, numerical or categorical properties are cast into different possible integer or floating-point precisions. The structure of the raw mesh binary object is itself embedded into enriched formats, for which a few standards exist, as RESQML™. The latter also encompasses structural information required to exchange models, generating information overhead. Finally, detail simplification through multiscale decompositions does not possess well-established quality metrics. All of the above hampers exhaustive objective evaluations such as possible in image processing, where metrics and benchmarks have been evaluated for decades.

To evaluate the performance of HexaShrink, we base our analysis on a set of seven geological meshes, with geometries ranging from smooth to fractured, and diverse categorical and continuous properties. Their main characteristics and properties are summarized in Table 1. As will be seen, they appear representative enough to allow one to derive consistent observations and conclusions for different data handling purposes.

They are initially stored in the GRDECL ("GRiD ECLipse") file format [46]. Originally complex geomodels are thus described with details rendering their geometry explicit and structured, an important feature for geomodelers or flow simulation software (Petrel™, SKUA-GOCAD™, Eclipse™...).

As observed in the state-of-the-art (Section 3.2), to our knowledge, reversible multiscale representations of geometry and properties—together with discontinuity preservation—of hexahedral meshes do not exist. Even if the standardized $3D$ extension to the JPEG2000 image compression format (termed J2K-3D) could process the properties as volumetric images, it is not *per se* suited to volume meshes, especially with categorical properties. We thus focus on visualizations, comparisons with geomodeler upscaling capabilities, and the embedding of our multiscale decompositions into several all-purpose compression algorithms.

The evaluation methodology is twofold. First, we exemplify the outcome of HexaShrink on meshes on

Table 1 Meshes chosen for evaluation: a compendium of their ontological characteristics and geological properties

Mesh	Characteristics				Properties		
index	# cells	Dimension	Faults	File size	ACTNUM	Continuous	Categorical
1	93,600	80 × 45 × 26	No	4.615 MB	100%	Porosity	Rock type
2	1,000,000	100 × 100 × 100	No	42.459 MB	100%	—	—
3	36,816	59 × 39 × 16	Yes	1.458 MB	100%	—	—
4	210,000	100 × 100 × 21	Yes	7.881 MB	20%	—	—
5	450,576	149 × 189 × 16	Yes	22.730 MB	46%	Porosity, permeability	—
6	5,577,325	227 × 95 × 305	Yes	274,573 MB	97%	Porosity	Rock type
7	13,947,600	240 × 295 × 197	Yes	580.937 MB	100%	Porosity	Rock type

either their geometry with a continuous and a categorical property at different dyadic scales. This reversible framework is put into perspective with similar downscaling processes in a reference geomodeler. Second, a comprehensive evaluation of lossless compression performance is provided, using state-of-the-art coders on either the raw files or their multiscale decomposed counterparts.

5.2 Reversible multiscale mesh representation

Figures 17 and 18 present the reversible multiscale decompositions generated by HexaShrink for mesh#1 and mesh#7. The latter contains several faults. Downsampled meshes are arranged in rows by decreasing scale. The first column represents the mesh without any attribute. The second and the third columns represent the same mesh onto which a continuous and a categorical property is mapped, respectively.

Mesh#1 is decomposed to the lowest possible resolution (Fig. 17, bottom). This is probably not useful from a geologist perspective. However, while all properties are almost constant, the lower arch corresponding to an anticlinal on the mesh at original resolution remains perceptible on the final "Lego brick" resolution. Looking at the porosity property (middle column), one observes how the values are progressively homogenized on coarser hexes. Concerning the rock type (last column), one observes that the modelet scheme tends to locally maintain predominant categories resolution after resolution, which is very satisfactory.

On Fig. 18, the much larger mesh#7 is represented down to a fifth dyadic sub-scale. It contains an isolated fault on the left side (the diagonal crest shape) and a faulty block on the right. Even at the coarsest level, corresponding to a downsampling by $2^4 \times 2^4 \times 2^4$, these two structural discontinuities are still present, while keeping a good shape fidelity, globally. Concerning the attributes, the decompositions are also adequate.

To emphasize further the capacity of our scheme to maintain coherency across the resolutions for the properties, Fig. 19 shows the evolution of the *Rock Type* distribution until the third resolution for mesh#1 and mesh#7. We observe that HexaShrink preserves the shape of histograms. In other words, the proportion of each category remains consistent across the scales of observation. Figure 19 also provides a comparison with the distributions obtained from the reversible CDF 5/3 wavelet transform (Section 4.2.3) used in J2K-3D [47]. One observes that histograms at lower resolution absolutely do not reflect the original ones, creating interpolated categorical values that do not possess geological meaning. Indeed, a major feature of HexaShrink is to combine, in an overall multiresolution framework, four different downsampling schemes adapted to each property.

Beyond these results in terms of geometry and property coherence across resolutions, we recall that our method is deterministic, and exact. The four analysis and synthesis multiresolution schemes allow perfect reconstruction. Contrary to [34], our method is able to manage all the fault configurations. HexaShrink is also scalable, which is indispensable in geosciences, given the steadily growing size of data volumes and model simulations. This scalability relies on an out-of-core algorithm [48] that splits geometry and property matrices into "small" sub-matrices, to process them sequentially. We are thus able to deal with meshes of any size. Lastly, GPU-based parallel computing have been also included, to speed the algorithm up.

As HexaShrink proposes a comprehensive reversible multiscale framework with dyadic downsampling, we compare it to related upscaling features for geomodels.

5.3 Geomodel upscaling: SKUA-GOCAD™ vs HexaShrink

SKUA-GOCAD™ or PETREL™ are frequently used in geosciences to handle geological objects and to generate meshes for flow simulation. These specific meshes describe

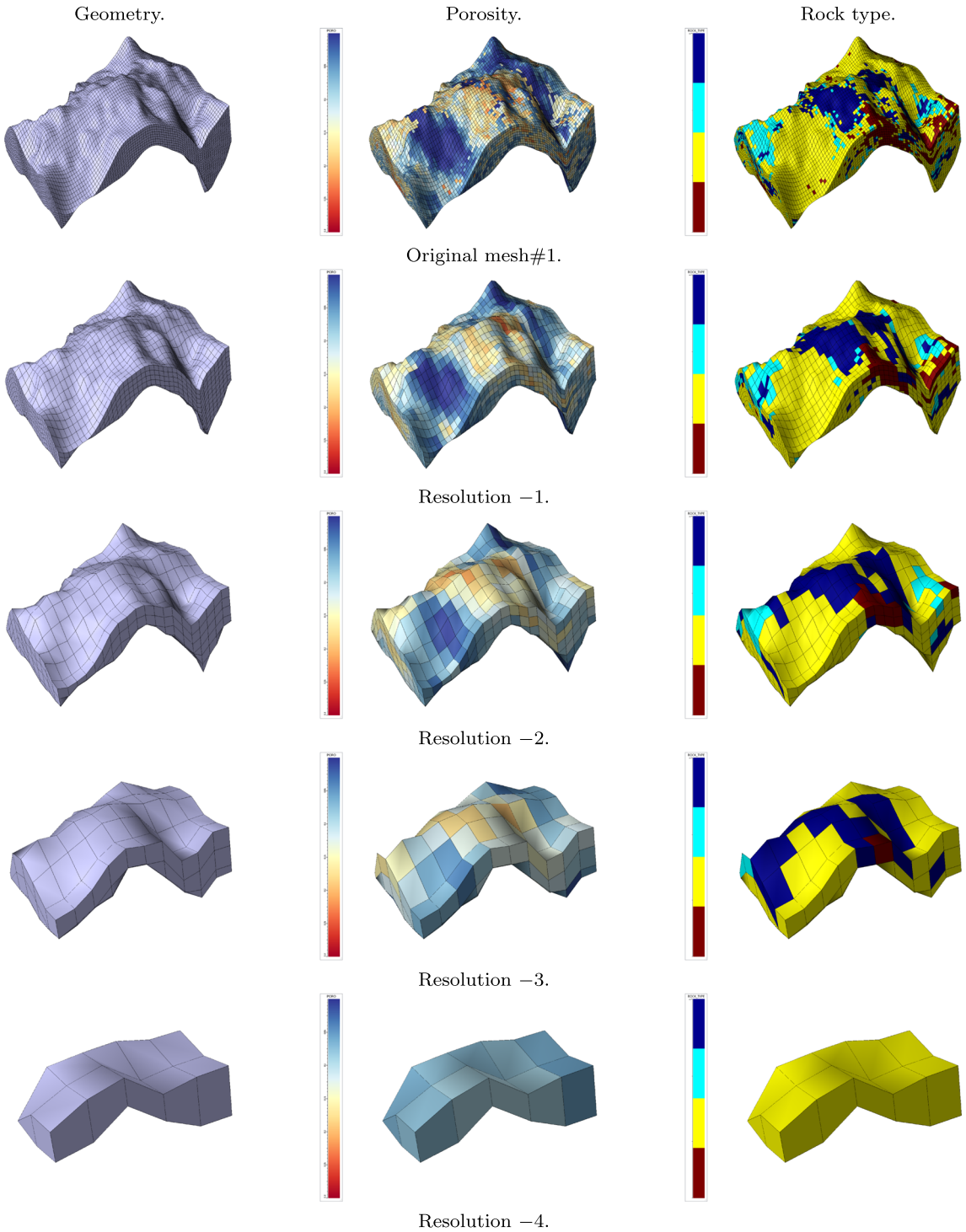


Fig. 17 Original mesh#1, its attributes, and four levels of resolution generated with HexaShrink

structural discontinuities whose impact is significant on simulation. To obtain such meshes, the geomodel—a surface description of horizons or faults—is fitted into a grid at the desirable resolution. Pillar orientation is influenced by fault dip and cell layer thickness is adapted to the distance between horizons. Additional properties can then be assigned to mesh cells: porosity, saturation, rock type…from well data or geological interpretation. Would one wish to lower the resolution, the process described above should be reiterated.

A simpler alternative proposed by geomodelers consists in *upscaling* meshes. Such methods are usually flexible

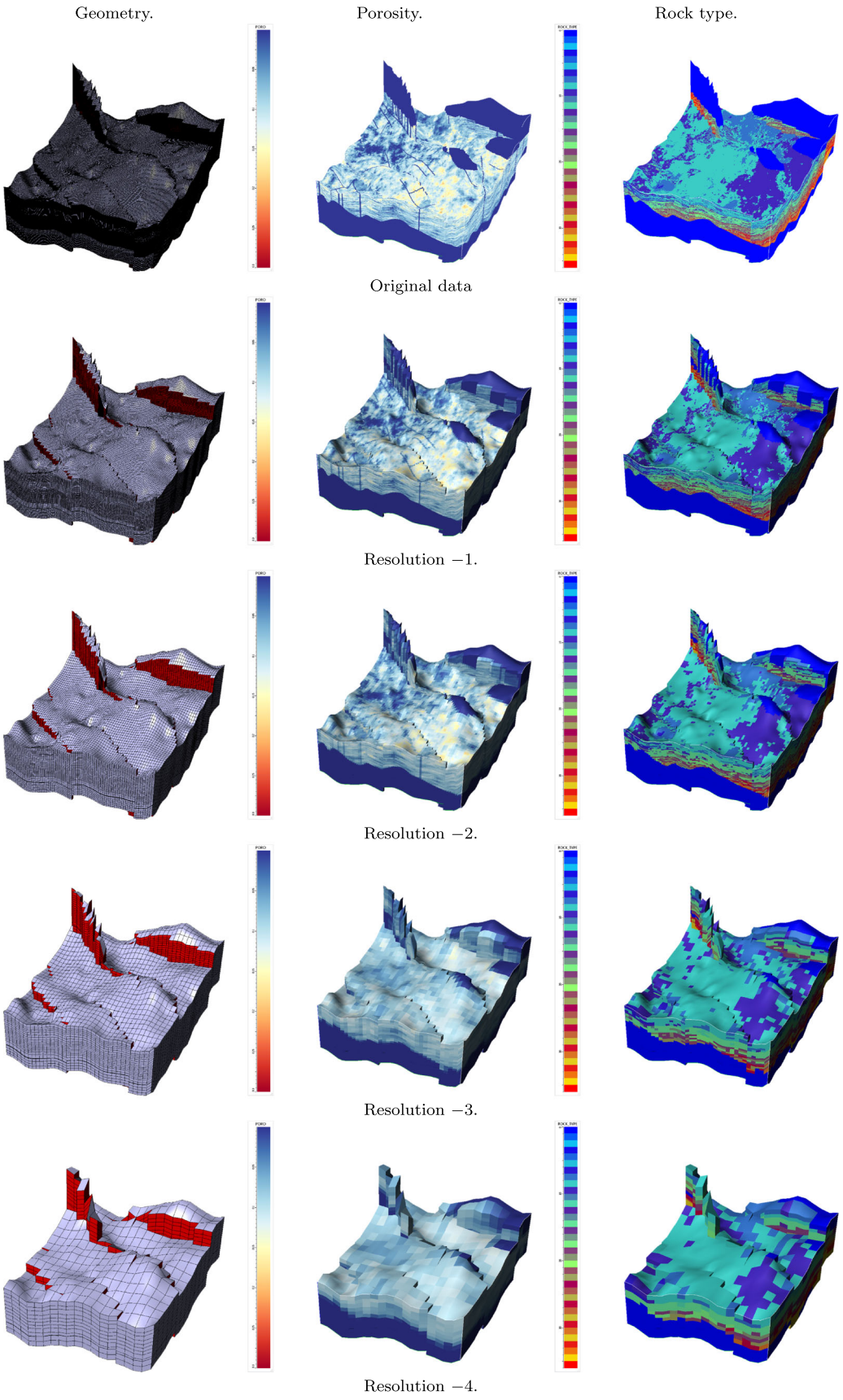


Fig. 18 Original mesh#7, its attributes, and four levels of resolution generated with HexaShrink

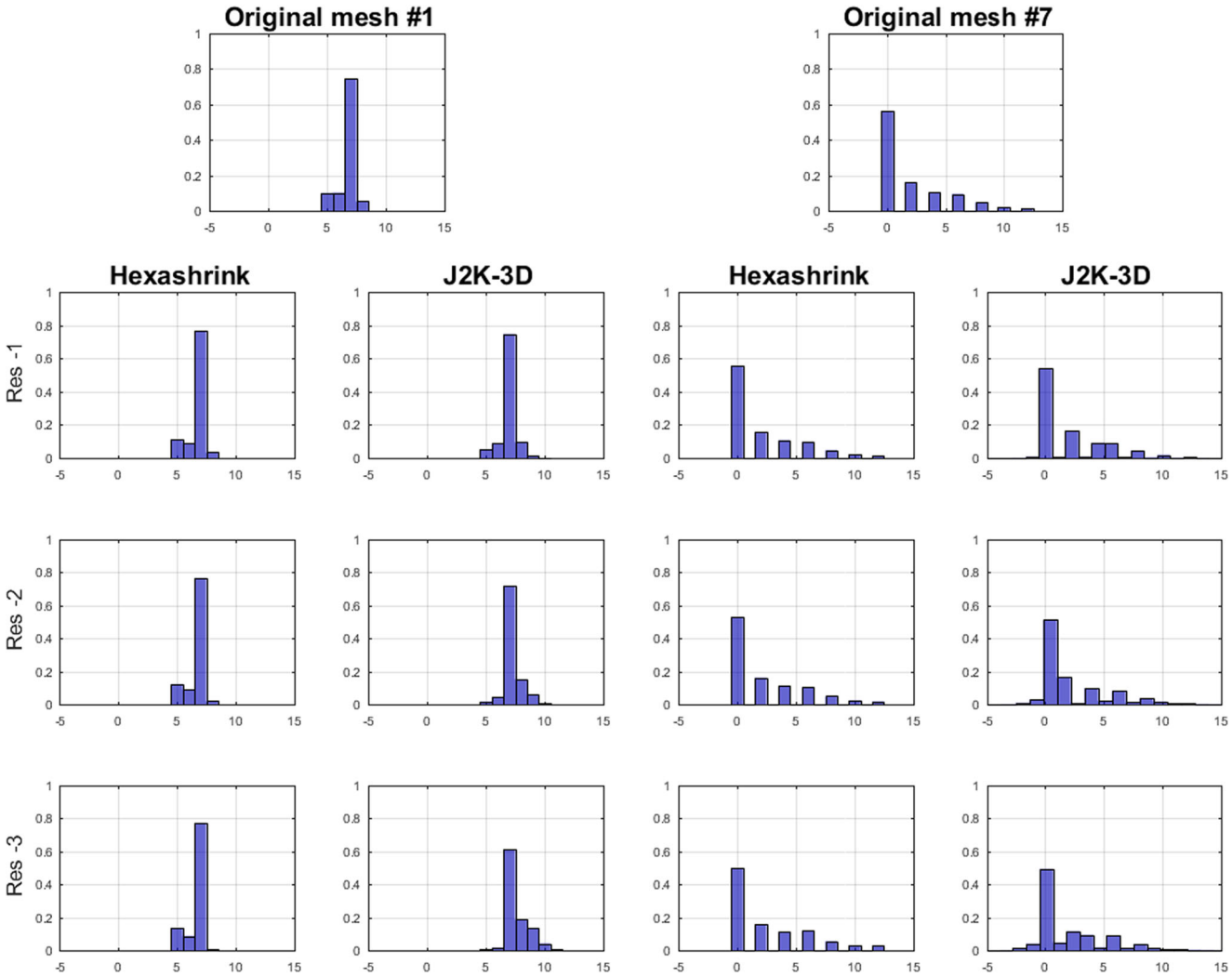


Fig. 19 Evolution of the distribution of the *Rock type* categories across resolutions for mesh#1 and mesh#7, decomposed with either HexaShrink's modelet or the rounded lifting CDF 5/3 used in the lossless J2K-3D

yet often *ad hoc*, converting geometry and properties in a non-reversible manner. Figure 20 confronts meshes #5 and #6 downsampled at power-of-two resolutions with SKUA-GOCAD™ and HexaShrink. HexaShrink tends to better preserve faults (colored in red), as compared to SKUA-GOCAD™. Figures emphasize an improved preservation of mesh borders, with an efficient management of ACTNUM throughout resolutions. Some artifacts may appear with SKUA-GOCAD™'s upscaling, which are automatically averted by HexaShrink, leading to nicer meshes at low resolution. As a summary, HexaShrink, while being fully reversible at dyadic scales only, efficiently and automatically manages structural discontinuities in the VMs. It may provide an interesting complement to existing irreversible upscaling proposed by several geomodelers.

5.4 Compression performance comparison

We now provide an objective evaluation of the HexaShrink multiscale representation for compression purposes. Our main objective is to verify that binary mesh formats (beyond mere ontological analyses) are indeed compressible, and whether decomposing them in a progressive manner over different scales remains beneficial in size reduction for needs beyond mere visualization (data storage, transfer). Indeed, multiscale representations are thought to enhance the sparsity of locally regular data, often resulting in better predicted properties that can subsequently be compressed.

We first assess lossless (or perfect) compression. All mesh ontological and geological (cf. Table 1, Section 5.1) are thus perfectly restored, whatever the number of decomposition levels (Section 5.2). Since our mesh objects are

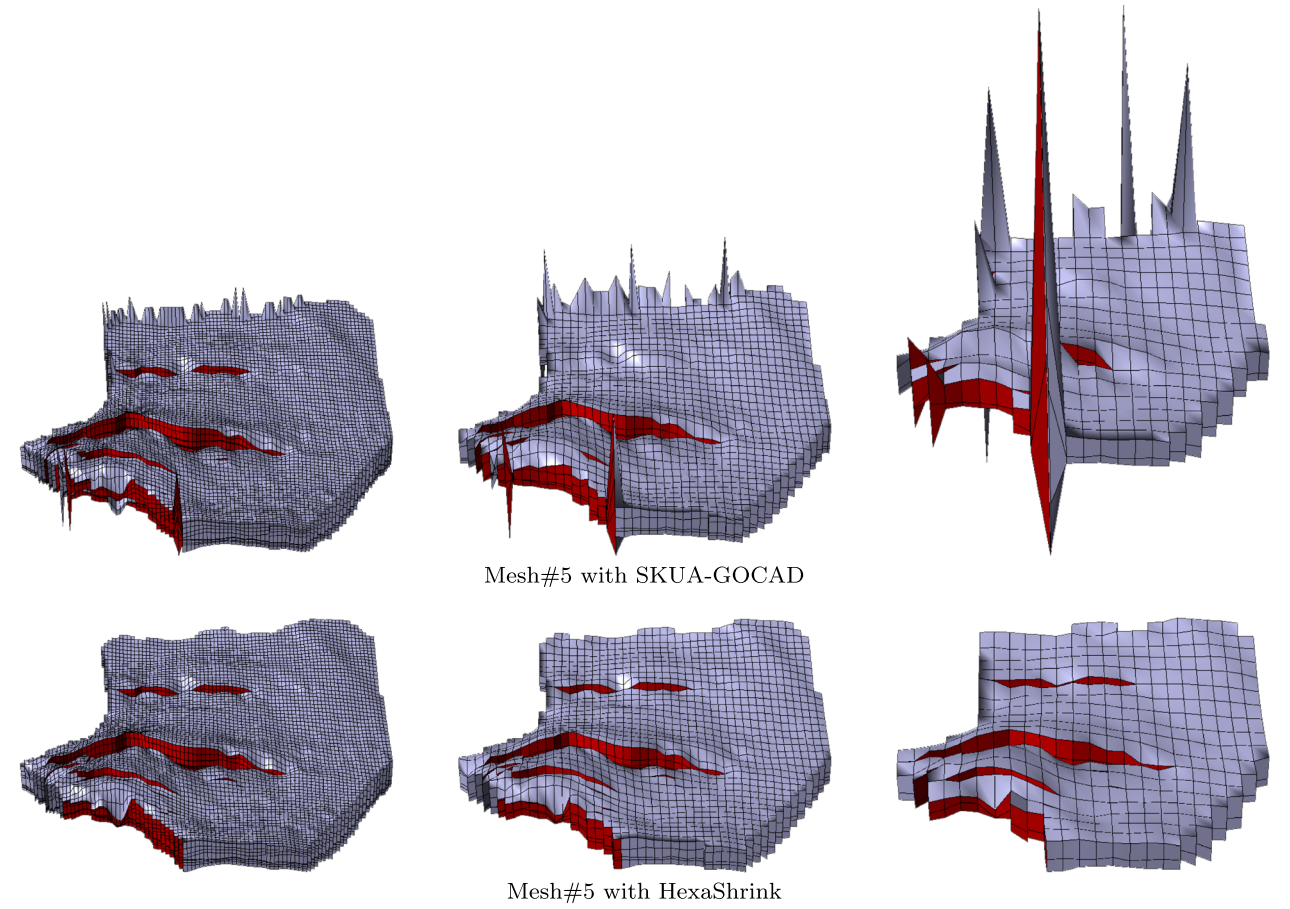


Fig. 20 After dyadic downsampling/upscaling, HexaShrink (bottom) better preserves faults, and manages non-active cells (i.e., with null ACTNUM values) across scales, yielding nicer borders at each resolution, contrary to GOCAD. From left to right: resolution − 1, − 2, and − 3, respectively

heterogeneous, we treat geometry and properties independently, and compress individually their approximations and details.

We compare three generations of lossless all-purpose encoders: gzip (1992), bzip2 (1996), LZMA (1998). They use Lempel-Ziv, Burrows-Wheeler, Lempel-Ziv-Markov entropic coding, respectively. We refer to [12, 49] for details regarding these state-of-the-art compression methods.

We exhaustively compare compression performances in Table 2. For the sake of clarity, recall that different computational methodologies exist: a compression ratio of 4 is given by the fraction between the sizes of the original file and the compressed one (the larger the ratio, the better the compression). The latter can also be related to its inverse, the smallest file representing 25% of the raw data ($\frac{1}{4} \times 100$), or a compression gain due to the reduction in size of 75% (corresponding to $\left(1 - \frac{1}{4}\right) \times 100$).

As an example, we provide a detailed interpretation of the third row, corresponding to the mildly complicated and faulty mesh#3. Without decomposition, i.e., by directly compressing the binary formats, gains in file size are already observed, from 62.5% ($\left(1 - \frac{1}{2.67}\right) \times 100$) for gzip to + 72.5% for LZMA. We first remark that improvements sensitively increase as we use more recent entropic coders, with only one exception for mesh#6, gzip performing slightly better than bzip2. However, the most recent LZMA coder always offers the best performances, with a sufficient gap over the two other methods.

The same trend applies when performing a one-level HexaShrink transformation on mesh#3, with an additional gain in compression: for instance, the combination of a 1-level HexaShrink associated with LZMA yields a compressed mesh twice as small (81.9%) as a direct gzip compaction on the original mesh. This is advantageous, as the proposed method either provides access to a twofold downsampled mesh together with a smaller overall size.

One can wish to have access to further levels of approximation. The third line of each table block specifies the range of additional available levels (depending on mesh size), here from 2 to 4, with the minimum and maximum compression ratios attained. While we still observe a marginal improvement over a 1-level HexaShrink (and again a slightly anomalous behavior for mesh#5), what is more important is the almost imperceptible variation between the resolutions. Hence, HexaShrink offers in all cases an interesting compression ratio with the supplementary interest of getting all intermediate resolutions, as shown earlier in Section 5.1.

Overall, the most basic worse case performance (with gzip) of HexaShrink provides a gain above 60% in size, which could be further exploited with hardware acceleration [50]. Or in the best cases, combined with LZMA, one can expect as much as 3.64–13.35 fold compression.

In rare cases, HexaShrink does not result in clear compression improvement. For mesh#7 and LZMA, we even observe that the best compression ratio (12.52) is obtained without HexaShrink decomposition. With such human or computer-generated objects, by contrast with natural data, this often stems from the inherent quantization of values. This typically happens in mesh#7 when intermediate coordinates, or properties, are obtained by interpolation, to refine geological layers. Variables with apparent high-dynamic range and precision, unbeknown to the user, may derive from easy-to-store indices. Floating-point depth coordinates may result from a mere affine relationship on a list $\mathcal{I}$ of small integers, with offset o and scale s: $s \times \mathcal{I} + o$. LZMA's superior capability owes to its capacity to capture complex models of byte patterns. By contrast, with a wavelet decomposition, the affine relationship in such a case is poorly captured throughout approximations, due to the rounding in wavelet lifting (Section 4.2.3). Hence, multiscale decompositions may slightly reduce the raw compression performance for meshes presenting initial "numerical format" artifacts or illusory floating-point precision.

Table 2 Comparative lossless coding performances with compression ratios at different HexaShrink resolution levels combining HexaShrink with gzip, bzip2, and LZMA

Mesh	Level	gzip	bzip2	LZMA
1	None	3.73	4.98	6.43
	1	5.62	6.07	7.52
	2–4	5.67	6.12–6.13	7.42–7.44
2	None	3.23	8.41	10.12
	1	6.49	10.82	11.81
	2–6	7.48–7.58	12.75–13.03	13.35
3	None	2.67	2.99	3.63
	1	3.88	4.70	5.24
	2–4	4.03–4.05	4.92–4.93	5.47–5.48
4	None	1.83	1.89	2.21
	1	2.64	3.06	3.48
	2–4	2.76	3.22–3.23	3.64–3.65
5	None	2.46	2.55	3.33
	1	3.14	2.83	3.71
	2–4	3.25–3.26	2.91–2.92	3.80–3.81
6	None	2.31	2.25	3.04
	1	3.31	3.53	4.44
	2–6	4.14–4.24	4.48–4.68	5.54–5.73
7	None	3.20	5.98	12.52
	1	5.42	7.07	8.90
	2–7	5.80–6.72	7.63–10.12	9.05–10.23

This however does not hamper the usability of HexaShrink for storage and visualization, as the direct access to a hierarchy of resolutions respecting discontinuities is granted, while already providing impressive compression rates of about 8–9, superior to most results for the others meshes.

Speed performance is highly dependent on mesh size, discontinuity complexity, levels of details. For a baseline evaluation, a Java implementation was run on a laptop with Intel Core i7-6820HQ CPU @ 2.70 GHz processor and 16 GB RAM. Each mesh (from our dataset in Table 2) was compressed to the maximum level, and decompressed, twelve times. As the outcomes were relatively stable, they were averaged. Timings for analysis or synthesis alone, and cumulated with lossless encoding and decoding, are summarized in Table 3.

Analysis is slightly slower than synthesis, both taking from a couple of seconds to a couple of minutes. Concerning the coders, gzip is the fastest, adding little overhead to HexaShrink speed. However, bzip2 or LZMA durations can reveal more expensive than analysis alone. The largest mesh is compressed in a little less than 13 m. What is more appealing in applications is that only a few seconds are necessary for small meshes. Lossless decompression is very fast, even for the largest mesh (13–20 s), as synthesis takes most of the time. And LZMA is the fastest here, adding only 5% overhead to synthesis, for recovering the whole original mesh. Decompressing lower resolutions only is of course even faster. This is interesting, as here we obtain a beneficial asymmetrical compression-decompression scheme, termed "compress once, decompress many." Once a model is built, one can afford to compress it only once, whatever the time it takes. Then, after transferring, handling it with the benefit of the smaller sizes, decompressing it many times is less expensive. The above performance could be greatly improved with more involved acceleration techniques.

As a result, on all tested examples, we demonstrate the possibility of storing independently multiscale mesh properties as approximations and details, while preserving geometry (hence faults). This method additionally has an important benefit in data handling, visualization and compression, all at a reasonable computational cost.

Table 3 Cumulative lossless compression and decompression durations of HexaShrink with gzip, bzip2, and LZMA, in seconds

	[A]nalysis	[A]+gzip	[A]+bzip2	[A]+LZMA
Min.	2.80	3.06	5.69	6.34
Max.	320	354	374.3520	760
	[S]ynthesis	[S]+gzip	[S]+bzip2	[S]+LZMA
Min.	0.79	1.03	3.35	3.30
Max.	264	280	284	277

6 Conclusion and perspectives

HexaShrink offers a comprehensive and efficient framework for a scalable representation of hexahedral meshes with continuous and categorical properties, at dyadic resolutions. It is first dedicated to the visualization of massive structured meshes, as used in geosciences, that can contain geometrical discontinuities, to describe faults for instance. Four adapted multiresolution representations are matched to the underlying nature of each data field. They permit to decompose such specific meshes progressively. In particular, this framework includes a morphological transform that takes into account geometrical discontinuities relative to any fault configuration, and preserve their rendering across resolutions, while maintaining structural coherency.

HexaShrink can process any mesh size, thanks to a GPU-based out-of-core algorithm. This is crucial, given the constant evolution of data acquisition density that yields increasingly massive and accurate dataset, associated to more demanding simulations. The HexaShrink decomposition is consistent with respect to mesh rescaling in geosciences, and provides an option for a reversible upscaling; [51] recently proposed such a wavelet-inspired scheme. Finally, it lends itself to an efficient lossless compression, which can be used for storage and transfer.

Perspectives can deploy into many directions. Motivated by preliminary progressive lossless compression results, we aim at developing a more versatile multiresolution compression scheme, to manage the potential evolution of mesh geometry or properties over time, with a special care for simulation-related quality metrics. As multiresolution analysis and synthesis were computed independently from compression, we also envision a better matched coding of approximations and details, for additional performance, toward lossy compression [52, 53].

Acknowledgments The authors would like to thank C. Dawson and M. F. Wheeler for their help, as well as the reviewers whose comments helped improve the compression performance assessment and comparison.

References

1. Peyrot, J.-L., Duval, L., Schneider, S., Payan, F., Antonini, M.: (H)exashrink: Multiresolution compression of large structured

hexahedral meshes with discontinuities in geosciences. In: Proc. Int. Conf. Image Process., pp. 1101–1105. Phoenix (2016)
2. Kober, C., Müller-Hannemann, M.: A case study in hexahedral mesh generation: simulation of the human mandible. Eng. Comput. **17**(3), 249–260 (2001)
3. Owen, S.J., Brown, J.A., Ernst, C.D., Lim, H., Long, K.N.: Hexahedral mesh generation for computational materials modeling. Procedia Eng. **203**, 167–179 (2017)
4. Cannon, S.: Reservoir Modelling: A Practical Guide. Wiley (2018)
5. Caumon, G., Gray, G., Antoine, C., Titeux, M.-O.: Three-dimensional implicit stratigraphic model building from remote sensing data on tetrahedral meshes: theory and application to a regional model of La Popa basin, NE Mexico. IEEE Trans. Geosci. Remote Sens. **51**(3), 1613–1621 (2013)
6. Lie, K.-A., Møyner, O., Natvig, J.R., Kozlova, A., Bratvedt, K., Watanabe, S., Li, Z.: Successful application of multiscale methods in a real reservoir simulator environment. Computat. Geosci. **21**(5-6), 981–998 (2017)
7. Perrons, R.K., Jensen, J.W.: Data as an asset: what the oil and gas sector can learn from other industries about "big data". Energy Pol. **81**, 117–121 (2015)
8. Cannon, S.: Simulation and Upscaling, pp. 181–204. Wiley (2018)
9. Dupont, F., Lavoué, G., Antonini, M.: 3D mesh compression. In: Lucas, L., Loscos, C., Remion, Y. (eds.) 3D Video from Capture to Diffusion. Wiley-ISTE (2013)
10. Røe, P., Hauge, R.: A volume-conserving representation of cell faces in corner point grids. Computat. Geosci. **20**(3), 453–460 (2016)
11. Lie, K.-A.: An Introduction to Reservoir Simulation Using MATLAB. User Guide for the Matlab Reservoir Simulation Toolbox (MRST). SINTEF ICT, Departement of Applied Mathematics (2016)
12. Salomon, D., Motta, G.: Handbook of Data Compression. Springer (2009)
13. Szymczak, A., Rossignac, J.: Grow & fold: Compressing the connectivity of tetrahedral meshes. Comput. Aided Des. **32**(8-9), 527–537 (2000)
14. Rossignac, J.: Edgebreaker: Connectivity compression for triangle meshes. IEEE Trans. Visual Comput. Graph. **5**(1), 47–61 (1999)
15. Gumhold, S., Straßer, W.: Real time compression of triangle mesh connectivity. In: Proc. SIGGRAPH Int. Conf. Comput. Graph. Interactive Tech., pp. 133–140 (1998)
16. Gumhold, S., Guthe, S., Straßer, W.: Tetrahedral mesh compression with the cut-border machine. In: Proc. IEEE Visualization Conf., pp. 51–58 (1999)
17. Isenburg, M., Alliez, P.: Compressing hexahedral volume meshes. Graph. Model. **65**(4), 239–257 (2003)
18. Touma, C., Gotsman, C.: Triangle mesh compression. In: Proc. Graphics Interface Conf., pp. 26–34. Vancouver, (1998)
19. Witten, I.H., Neal, R.M., Cleary, J.G.: Arithmetic coding for data compression. Commun. ACM **30**(6), 520–540 (1987)
20. Krivograd, S., Trlep, M., Žalik, B.: A hexahedral mesh connectivity compression with vertex degrees. Comput. Aided Des. **40**(12), 1105–1112 (2008)
21. Lindstrom, P., Isenburg, M.: Lossless compression of hexahedral meshes. In: Proc. Data Compression Conf., pp. 192–201 (2008)
22. Ibarria, L., Lindstrom, P., Rossignac, J.: Spectral predictors. In: Proc. Data Compression Conf., pp. 163–172 (2007)
23. Chen, D., Chiang, Y.-J., Memon, N., Wu, X.: Geometry compression of tetrahedral meshes using optimized prediction. In: Proc. Eur. Sig. Image Proc. Conf., pp. 4–8 (2005)
24. Isenburg, M., Lindstrom, P., Snoeyink, J.: Streaming compression of triangle meshes. In: Proc. Eurographics Symp. Geom. Process., vol. 255, pp. 111–118 (2005)
25. Isenburg, M., Lindstrom, P., Gumhold, S., Shewchuk, J.: Streaming compression of tetrahedral volume meshes. In: Proc. Graphics Interface, pp. 115–121 (2006)
26. Courbet, C., Isenburg, M.: Streaming compression of hexahedral meshes. Vis. Comput. **26**(6-8), 1113–1122 (2010)
27. Pajarola, R., Rossignac, J., Szymczak, A.: Implant sprays: Compression of progressive tetrahedral mesh connectivity. In: Proc. IEEE Visualization Conf., pp. 299–305 (1999)
28. Staadt, O.G., Gross, M.H.: Progressive tetrahedralizations. In: Proc. IEEE Visualization Conf., pp. 397–402 (1998)
29. Hoppe, H., DeRose, T., Duchamp, T., McDonald, J., Stuetzle, W.: Mesh optimization. In: Proc. ACM SIGGRAPH Comput. Graph., pp. 19–26 (1993)
30. Danovaro, E., De Floriani, L., Lee, M.T., Samet, H.: Multiresolution tetrahedral meshes: an analysis and a comparison. In: Proc. Shape Modeling International, pp. 83–91 (2002)
31. Jacques, L., Duval, L., Chaux, C., Peyré, G.: A panorama on multiscale geometric representations, intertwining spatial, directional and frequency selectivity. Signal Process. **91**(12), 2699–2730 (2011)
32. Boscardín, L.B., Castro, L.R., Castro, S.M., Giusti, A.D.: Wavelets bases defined over tetrahedra. INSTEC J. Comput. Sci. Technol. **6**(1), 46–52 (2006)
33. Bey, J.: Tetrahedral grid refinement. Computing **55**(4), 355–378 (1995)
34. Chizat, L.: Multiresolution Signal Compression: Exploration and Application. M.S. thesis, ENS Cachan (2014)
35. Chaux, C., Pesquet, J.-C., Duval, L.: Noise covariance properties in dual-tree wavelet decompositions. IEEE Trans. Inform. Theory **53**(12), 4680–4700 (2007)
36. Chaux, C., Duval, L., Benazza-Benyahia, A., Pesquet, J.-C.: A nonlinear Stein based estimator for multichannel image denoising. IEEE Trans. Signal Process. **56**(8), 3855–3870 (2008)
37. Sweldens, W.: The lifting scheme: A custom-design construction of biorthogonal wavelets. Appl. Comput. Harmon. Analysis **3**(2), 186–200 (1996)
38. Bruekers, F.A.M.L., van den Enden, A.W.M.: New networks for perfect inversion and perfect reconstruction. IEEE J. Sel. Areas Comm. **10**(1), 129–137 (1992)
39. Rao, R.M., Bopardikar, A.S.: Wavelet Transforms: Introduction to Theory and Applications. Prentice Hall (1998)
40. Kovačević, J., Goyal, V., Vetterli, M.: Signal Processing Fourier and Wavelet Representations (2012)
41. Rezapour, A., Ortega, A., Sahimi, M.: Upscaling of geological models of oil reservoirs with unstructured grids using lifting-based graph wavelet transforms. Transp Porous Med. (2019)
42. Le Gall, D., Tabatabai, A.: Sub-band coding of digital images using symmetric short kernel filters and arithmetic coding techniques. In: Proc. Int. Conf. Acoust. Speech Signal Process., pp. 11–14 (1988)
43. Cohen, A., Daubechies, I., Feauveau, J.-C.: Biorthogonal bases of compactly supported wavelets. Commun. ACM **45**(5), 485–560 (1992)
44. Calderbank, A.R., Daubechies, I., Sweldens, W., Yeo, B.-L.: Wavelet transforms that map integers to integers. Appl. Comput. Harmon. Analysis **5**(3), 332–369 (1998)
45. Antonini, M., Payan, F., Schneider, S., Duval, L., Peyrot, J.-L.: Method of exploitation of hydrocarbons of an underground formation by means of optimized scaling. Patent (2017)
46. Pettersen, Ø.: Basics of reservoir simulation with the Eclipse reservoir simulator. Department of Mathematics, University of Bergen, Norway. Lecture Notes (2006)
47. ITU-T T.809: JPEG2000 image coding system: Extensions for three-dimensional data. ISO/IEC 15444-10:2011 (2011)

48. Isenburg, M., Gumhold, S.: Out-of-core compression for gigantic polygon meshes. In: Proc. SIGGRAPH Int. Conf. Comput. Graph. Interactive Tech., pp. 935–942 (2003)
49. Nelson, M., Gailly, J.-L.: The Data Compression Book. Wiley (1995)
50. Abdelfattah, M.S., Hagiescu, A., Singh, D.: Gzip on a chip: High performance lossless data compression on FPGAs using OpenCL. In: Proc. Int. Workshop OpenCL, pp. 12–13 (2014)
51. Misaghian, N., Assareh, M., Sadeghi, M.: An upscaling approach using adaptive multi-resolution upgridding and automated relative permeability adjustment. Computat. Geosci. **22**(1), 261–282 (2018)
52. Lu, T., Liu, Q., He, X., Luo, H., Suchyta, E., Choi, J., Podhorszki, N., Klasky, S., Wolf, M., Liu, T., Qiao, Z.: Understanding and modeling lossy compression schemes on HPC scientific data. In: IEEE International Parallel and Distributed Processing Symposium, pp. 21–25 (2018)
53. Liang, X., Di, S., Tao, D., Li, S., Li, S., Guo, H., Chen, Z., Cappello, F.: Error-controlled lossy compression optimized for high compression ratios of scientific datasets. In: IEEE Int. Conf. Big Data, pp. 10–13 (2018)

Affiliations

Jean-Luc Peyrot[1] · Laurent Duval[2,3] · Frédéric Payan[4] · Lauriane Bouard[1] · Lénaïc Chizat[2,5] · Sébastien Schneider[1,6] · Marc Antonini[4]

Jean-Luc Peyrot
Jlucp@laposte.net

Frédéric Payan
fpayan@i3s.unice.fr

Lauriane Bouard
Lauriane.Bouard@ifpen.fr

Lénaïc Chizat
Lenaic.Chizat@inria.fr

Sébastien Schneider
Sebastien.Schneider@holomake.fr

Marc Antonini
am@i3s.unice.fr

1 IFP Energies nouvelles, Rond-Point de l'échangeur de Solaize, BP3, F-69360 Solaize, France
2 IFP Energies nouvelles, 1 et 4 avenue de Bois-Préau, F-92852 Rueil-Malmaison, France
3 ESIEE Paris, University Paris-Est, LIGM, F-93162 Noisy-le-Grand, France
4 CNRS, I3S, Université Côte d'Azur, 2000, route des Lucioles, F-06900 Sophia Antipolis, France
5 Present address: INRIA, ENS, PSL Research University Paris, Paris, France
6 Present address: HoloMake, Meudon, France