



Parallel homological calculus for 3D binary digital images

Fernando Díaz-del-Río¹ · Helena Molina-Abril² · Pedro Real¹  · Darian Onchis³ · Sergio Blanco-Trejo⁴

Accepted: 14 November 2023 / Published online: 29 January 2024
© The Author(s) 2024

Abstract

Topological representations of binary digital images usually take into consideration different adjacency types between colors. Within the cubical-voxel 3D binary image context, we design an algorithm for computing the isotopic model of an image, called **(6, 26)**-Homological Region Adjacency Tree (**(6, 26)**-*Hom-Tree*). This algorithm is based on a flexible graph scaffolding at the inter-voxel level called Homological Spanning Forest model (HSF). *Hom-Trees* are edge-weighted trees in which each node is a maximally connected set of constant-value voxels, which is interpreted as a subtree of the HSF. This representation integrates and relates the homological information (connected components, tunnels and cavities) of the maximally connected regions of constant color using 6-adjacency and 26-adjacency for black and white voxels, respectively (the criteria most commonly used for 3D images). The Euler-Poincaré numbers (which may as well be computed by counting the number of cells of each dimension on a cubical complex) and the connected component labeling of the foreground and background of a given image can also be straightforwardly computed from its *Hom-Trees*. Being I_D a 3D binary well-composed image (where D is the set of black voxels), an almost fully parallel algorithm for constructing the *Hom-Tree* via HSF computation is implemented and tested here. If I_D has $m_1 \times m_2 \times m_3$ voxels, the time complexity order of the reproducible algorithm is near $O(\log(m_1 + m_2 + m_3))$, under the assumption that a processing element is available for each cubical voxel. Strategies for using the compressed information of the *Hom-Tree* representation to distinguish two topologically different images having the same homological information (Betti numbers) are discussed here. The topological discriminatory power of the *Hom-Tree* and the low time complexity order of the proposed implementation guarantee its usability within machine learning methods for the classification and comparison of natural 3D images.

Keywords 3D digital images · Binary images · Parallel computing · Cavity · Tunnel · Connected component · Homological spanning forest · Inter-voxel · Homological region adjacency tree

The first three authors are supported by project PID2019-110455GB-I00 (AEI/FEDER,UE) and project US-1381077 (JJAA/FEDER, UE).

✉ Pedro Real
real@us.es

Extended author information available on the last page of the article

Mathematics Subject Classification (2020) 62H35 · 68W10 · 68Q10

1 Introduction

Nowadays, there is a strong interest in integrating computational topological notions and methods to enhance both classical machine learning algorithms and deep learning models [1, 2]. This interest is motivated, on the one hand, by the power of topological invariants and representations for capturing global intrinsic properties of data sets and, on the other, by their high degree of understandability and learning for both humans and computers.

A classic scheme in image understanding algorithms usually includes two important initial steps: Segmentation and region connectivity representation. The first one deals with partitioning the initial image into regions, and in the last one a new data structure is created specifying the connectivity relations between them. In turn, two fundamental types of connectivity representations are distinguished in digital image processing attending to the smallest component of the structure: those in which the n -xel is the minimum unit of connectivity information (adjacency information) and those that work at inter- n -xel level (incidence information). In adjacency-based representations, the most commonly used data structure is that of a graph (see, for example, [3]). These models depend on the type of adjacency used between n xels. For cubical- n -xel n -dimensional digital images, usually $2n$ and $(3^n - 1)$ -adjacencies are considered. For incidence-based representations, abstract cell complexes (ACC) are frequently considered as supporting data structures (see, for instance, [4]). In fact, since an nD digital image is naturally represented by an n -dimensional ACC and a graph can be seen as a 1-dimensional ACC, graph-based connectivity representations can be seen as strategies for simplifying topological information by dimensionality reduction. In this sense, classical region connectivity representations based on graphs may be considered as appropriate tools for extracting partial topological information from nD digital images (considered as ACCs) generally involving 0-dimensional holes (connected components) and $(n - 1)$ -holes (hypercavities).

One of the most notorious examples of graph-based connectivity representations is the region adjacency graph (RAG), which is based mainly on two connectivity properties among sets of pixels: adjacency and inclusion (see, for instance, [5, 6]). λ -RAG can be considered as a connectivity model using λ -adjacency between n -xels. For nD images based on square pixels, $2n$ -RAG and $(3^n - 1)$ -RAG are the most common representations. From a topological viewpoint, a region can be seen as an appropriate “cell complex hull” of the set of n -xels that forms such a region. If the color palette of the image consists of two values (black=1 and white=0), the pairs of different adjacency types $(2n, 3^n - 1)$ or $(3^n - 1, 2n)$ for black and white pixels are generally used for the RAG. For example, the nodes of an $(2n, 3^n - 1)$ -RAG (in fact, a RAG tree) represent ACCs (also called topological regions) generated by black $2n$ -connected components (CCs) and white $(3^n - 1)$ -CCs. A duality topological property is exhaustively used in nesting or inclusion relationships between regions: to identify an $(n - 1)$ -dimensional hole of a topological region generated by a λ -CC of one color with a 0-dimensional hole of a region generated by a λ' -CC of the other color that is “included” or “surrounded” by the first (with $\lambda, \lambda' \in \{2n, 3^n - 1\}, \lambda' \neq \lambda$). In the case of 3D binary images, the $(6, 26)$ -RAG tree and the $(26, 6)$ -RAG tree are the most common representations of region connectivity. The RAG representation for a set of well-known topological objects is shown in Fig. 1.

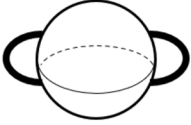

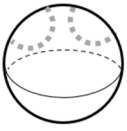


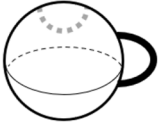


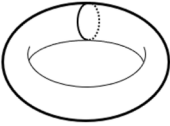

Three Spheres with inner or outer handles and a torus	RAG	Hom-Tree
		
		
		
		

Fig. 1 The RAG and the *Hom-Tree* representation of four simple objects immersed in a 3D digital image (white background): A sphere with two outer handles, a sphere with two inner handles, a sphere with one outer and one inner handle, and a torus

In [6], Chapter 2, Rosenfeld writes: “The topology of an n -dimensional $(2n, 3^n - 1)$ -picture is defined by its RAG Tree. Two pictures are topologically equivalent if their RAG Trees are isomorphic”. Restricted to binary 2D images, the RAG tree contains all the homotopy-type information of the foreground (black object according to our convention), but in higher dimensions, this is, in general, not true (see [7]). In [8], a new representation called *Hom-Tree* that improves the RAG-tree-based topological discrimination of 3D binary images is presented.

Definition 1 [8] Let I_D be a 3D binary image. The $(6, 26)$ -Homological Tree of I_D , denoted by $(6, 26) - HomTree(I_D)$, is an edge-weighted $(6, 26)$ -RAG Tree. The weight of an edge (R, S) , being R and S two nested regions of the *Hom-Tree* (and, consequently, one is necessarily a black 6-CC and the other one is a white 26-CC), is the number of 1-dimensional homological holes “shared” by R and S as ACCs.

Only partial topological calculations related to the *Hom-Tree* representation are provided in [8]. Here, we demonstrate that this unique representation can be efficiently computed from a complete and flexible homotopy model of I_D (considered as a cubical ACC inter-voxel level) called Homological Spanning Forest (or HSF, for short) (see [9, 10]).

The ability to isotopically differentiate two homologically equivalent volumes of the *Hom-Tree* representation is illustrated in the following. A simple example is shown in Fig. 1 with several digital spheres having handles in various placements. Figure 1 shows the continuous versions of these volumes and their respective *Hom-Trees*. Let us note that these volumes as ACCs have the same Betti numbers, or, in other words, they are homologically equivalent. All of them have one single connected component, two handles, and one cavity. In all these cases, their RAG coincides, but their *Hom-Tree* representations are different from each other. Let us note that the number of 1-dimensional homological holes shared by two regions is different from the number of tunnels of the digital frontier between them. These examples show how the proposed representation represents a step forward in improving the topological classification of 3D digital images.

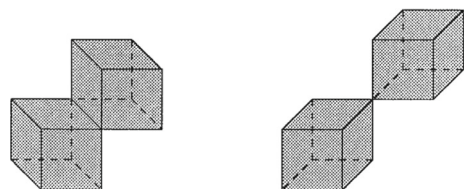
However, the ability to topologically recognize 3D binary images or simple volumes of the *Hom-Tree* is limited. For instance, given a sphere connected to two handles (one interior and one exterior) and a digital torus, both have the same *Hom-Trees* and they are actually topologically different 3D objects (see Fig. 1).

An almost fully parallel algorithm for computing the *Hom-Tree* of a well-composed binary image I_D is designed and implemented here. Let us now introduce the “well-composed” concept. A (black) volume D is well-composed if any 26-CCs of D is also its 6-CC. A 3D binary image I_D is well-composed if D is well-composed. In that case, (6, 26)-*Hom-Tree* coincides with (26, 6)-*Hom-Tree*. Figure 2 shows critical configurations of cubes (modulo reflections and rotations) that may not occur within D so that it is well-composed (see [11]).

The theoretical time complexity order of the proposed method is near $O(\log(m_1 + m_2 + m_3))$, being $m_1 \times m_2 \times m_3$ the dimensions of I_D . We divide here its computation in four different Algorithms 1–4, each one including a line in which its theoretical time complexity order is detailed. A parallel computational method for obtaining the Euler-Poincaré characteristic of D (both with 6-adjacency and 26-adjacency) via the construction of the (6, 26)-*Hom-Tree* of I_D is also highlighted. Finally, the potential applications of the *Hom-Tree* representation to 4D data in the field of computational and artificial intelligence are envisioned.

The paper is organized as follows. The state of the art is reviewed in Section 2. In Section 3 some basic concepts needed for understanding this paper and an overall description of the algorithm that extracts the HSF and the *Hom-Tree* are introduced. Section 4 fully describes the parallel implementation of the previous representations for well-composed images. Section 5 exposes some examples of *Hom-Tree* representations for synthetic images and shows some preliminary results and future applications within the medical image and machine learning context. The paper concludes in Section 6.

Fig. 2 Critical configurations of cubes (modulo reflections and rotations) that may not occur within a 3D volume so that it is well-composed



2 Related works

Some of the most relevant conceptual topological representations applied to digital images can be classified into three groups:

- Models based on graphs (region adjacency graphs, image pyramids, component tree, alpha trees, tree of shapes, image pyramid [12], etc.) ;
- Models based on combinatorial topology (discrete Morse theory [13], combinatorial maps [14], Reeb graphs [15], skeletons [16, 17], Homological Spanning Forest model [18, 19], etc.);
- Models based on algebraic topology (Effective Homology [20], Algebraic-Topological-model [21, 22], Persistent Homology [23, 24], etc.).

All of these models are constructed using an abstract, algebraic, or geometric scenario for the given digital image, called *embedding model*, allowing image topological calculus. The most common embedding models for digital images are based on polyhedral tessellations, lattices, graphs, and ACCs. Polyhedral representations of nD ($n \geq 2$) digital images based on cubical complex decomposition and subsequent descriptions of the RAG tree for the adjacency pair $(2n, 3^n - 1)$ are given, for instance, in [25–27]. The ACC embedding model for digital images has been extensively studied in [28].

Given a binary 3D image I_D (where D corresponds to the set of black voxels) and an adjacency pair (λ, λ') (being $\lambda, \lambda' \in \{6, 26\}$ and $\lambda \neq \lambda'$), a new edge-weighted graph structure called (λ, λ') -*Homological Region Adjacency Tree (Hom-Tree*, for short) was firstly theoretically defined in [8]. This structure, seen as an unweighted graph, coincides with the classical RAG tree and contains all the λ -homological information of the foreground D and all the λ' -homological information of the background, both defined by their respective analogous cell complexes. The present paper is the first work in which a parallel algorithm for computing the *Hom-Tree* of a well-composed binary 3D image is designed and fully implemented.

Let λ be 6 or 26. The λ -*Euler-Poincaré characteristic or number* of a binary 3D image I_D (fixing a priori the adjacency type λ for the foreground set of n-xels D) is one of the most important topological invariants in digital topology and many methods have been proposed for its computation. For a 3D image I_D , the λ -Euler-Poincaré number can be globally expressed by three Betti numbers [25], the formula is as follows:

$$\chi(I_D) = \beta_0 - \beta_1 + \beta_2$$

where, β_0 is the number of maximally λ -connected foreground set (called also *objects*), β_1 is the maximal number of nonseparating cuts (also called *tunnels*) and β_2 is the number of cavities or voids that the foreground object has. The Euler-Poincaré number can be locally calculated by the formula:

$$\chi(I_D) = n_0 - n_1 + n_2 - n_3$$

where n_k is the number of cells of dimension k ($k = 0, 1, 2, 3$) of any abstract cell complex “geometrically modeling” the foreground using λ -adjacency type. In the next section, cubical cell complexes analogous of this kind are described. Numerous methods of Euler-Poincaré number computation based on local measures are reported in the literature (see, for example, [29, 30]). On the other hand, the number of methods for computing Euler-Poincaré number based on Betti numbers is much smaller and is essentially based on homology calculation ([31]). From Definition 1 of the *Hom-Tree*, the following result immediately follows:

Theorem 1 [8] *Let I_D a binary 3D image. From the (6, 26)-Hom-Tree representation of I_D , it is possible to straightforwardly compute the 6-Euler-Poincaré number of I_D and the 26-Euler-Poincaré number of $I_{I \setminus D}$.*

3 Understanding topological calculus for digital images

The topological calculus proposed here rests on three abstract image models that respectively shape the input, the functional process and the output of any topological image processing operation: the embedding model, the functional model and the conceptual model. The embedding image model is determined by the notion of abstract cellular complexes, which naturally allow topological aspects to be worked on. The functional model is based on the concept of dynamical abstract cell complexes, which are nothing but the “wire skeleton” of the ACC input. And, finally, homological image information is straightforwardly attainable from any Homological Spanning Forest representation of the image. This nonunique conceptual model can be derived from the functional one thanks to an exhaustive process of erasing incidence relationships but preserving global connectivity at homological level.

3.1 Embedding and functional models

First, we provide a slightly modified version of the definition of the classical relational ACC notion (see [4] for a survey). Mainly, the ACC bounding relation is specified by a function for which the “transitivity” is not assumed.

Definition 2 The triple $K = (K, dms, B)$ is a *dynamical abstract cell complex* (DACC, for short) if it is endowed by the following elements:

- *Structure.* $K = \bigcup_{q \geq 0} K_q$ is a (non-negative integer) graded set of elements (cells), such that $K_p \cap K_q = \emptyset$, for $p \neq q$;
- *Dimension.* A dimension function $dms : K \rightarrow \mathbb{N} \cup \{0\}$ defined by $dms(c) = q$ for $c \in K_q$;
- *Function.* A bounding function $B : K \times K \rightarrow \mathbb{N} \cup \{0\}$, satisfying that if $B(c', c) \neq 0$, then $dms(c') = dms(c) - 1$.

In the case of considering a transitive bounding relation instead of a bounding function B , then we retrieve the notion of the classical ACC [32]. Given a DACC $K = (K, dms, B)$, it is possible to construct an ACC, denoted by $ACC(K) = (K, dms, \bar{B})$, having the same structure K and dimension dms as that of K , and with a transitive bounding relation generated by the rule: $(c', c) \in \bar{B}$ if $B(c', c) \neq 0$. Reciprocally, given an ACC $K = (K, dms, B)$, its associated DACC, that is, $DACC(K) = (K, dms, \bar{B})$, can be constructed having the same structure and dimension as that of K and a bounding function defined by $\bar{B}(c', c) = 1$ if $(c', c) \in B$ and $dms(c') = dms(c) - 1$, and zero otherwise. The following result follows immediately:

Proposition 1 *Let $K = (K, dms, B)$ be an ACC. The Euler-Poincaré characteristic of K coincides with that of $DACC(K)$.*

Let us define the *boundary set* $\partial_K(c)$ of a q -cell $c \in K_q$ as follows: $\partial_K(c) = \{c' \in K_{q-1} \mid B(c', c) = 1\}$. Analogously, we can define the *coboundary set* of a cell.

Definition 3 A geometric or homologically coherent $DACC = (K, dms, B)$ is a DACC such that: (a) If $B(c', c) \neq 0$, then $B(c', c) = 1, \forall c', c \in K$; and (b) $\sum_{c', c'' \in K} B(c'', c') * B(c', c)$ is an even number, $\forall c \in K$. Let us note that $*$ is the operation of the product for $\mathbb{N} \cup \{0\}$. A geometric ACC K is an ACC such that its associated $DACC(K)$, is geometric.

More concretely, we deal here with the classical geometric ACC analogous $Cell(I_D)$ of a 3D binary digital image I_D based on cubical voxels. $Cell(I_D)$ is an ACC in which the physical cubical voxels (volumes), their faces, edges and vertices are the different dimension cells of $Cell(I_D)$ (see [33]). Its bounding function counts the incidence relations between two (geometrical) cells. Its associated DACC, $DACC(Cell(I_D))$, can be expressed under a node-weighted graph format, called the *connectivity graph* $G(Cell(I_D))$ of I_D . It is a graph having as nodes the cells of $Cell(I_D)$ and as edges the un-ordered incidence relations between cells with dimensions differing in one.

The *topological coordinate system* used in this paper for the connectivity graph $G(Cell(I_D))$ of an image I_D having dimensions $m_1 \times m_2 \times m_3$, is based on a matrix encoding of I_D . Let us first define $(\mathbb{N} \cup \{0\}) + \frac{1}{2}$ as the set of rational numbers of the kind $n + \frac{1}{2}$, where $n \in \mathbb{N} \cup \{0\}$. The 3-cells (physical voxel) are encoded by the vectors (x_1, x_2, x_3) , $1 \leq x_i \leq m_i, x_i \in \mathbb{N}, i = 1, 2, 3$. The 2-cells are encoded by vectors (x_1, x_2, x_3) , where two coordinates are natural numbers, and the other one belongs to $(\mathbb{N} \cup \{0\}) + \frac{1}{2}$. The 1-cells are encoded by vectors (x_1, x_2, x_3) , where one coordinate is a natural number, and the other two belong to $(\mathbb{N} \cup \{0\}) + \frac{1}{2}$. Cells 0 are encoded by vectors (x_1, x_2, x_3) , where all coordinates belong to $(\mathbb{N} \cup \{0\}) + \frac{1}{2}$. Figure 5 (left) shows an example of such a system.

Depending on the type of adjacency chosen (6 or 26), given a set of voxels V of I_D , we define the following associated ACCs:

- The 26-adjacency cell complex hull $Cell_{26}(V) = (Cell_{26}(V), dim_{26}, B_{26})$, which is the ACC such that its 3-dimensional cells are the physical voxels of V and its 2, 1 and 0-cells are the corresponding faces, edges and corners of the voxels of V . Its dimension function applied to the cell c is denoted by $dim_{26}(c)$. Its bounding relation satisfies $(c', c) \in B_{26}$ iff c' has a lower dimension than c and they are incident.
- The 6-adjacency cell complex hull $Cell_6(V) = (Cell_6(V), dim_6, B_6)$ which might be seen as a “dual” ACC to $Cell_{26}(V)$. Here, the 0-dimensional cells are the physical voxels of V , its 1-cells are specified by a pair of 6-adjacent volumes, its 2-cells are 4-tuples of volumes (v_1, v_2, v_3, v_4) such that v_i and v_j ($i \leq j$) are 6-adjacent if $i - j = 3 \pmod 4$, and its 3-cells are 8-tuples of volumes forming a cube. Its dimension function applied to the cell c is denoted by $dim_6(c)$. Its bounding relation satisfies $(c', c) \in B_6$ iff c' has a lower dimension than c and they are incident.

Note that $Cell_6(V)$ is a cellular subcomplex of $Cell_{26}(V)$ (analogously, for the corresponding DACC grids) and that $Cell(I_D) = Cell_{26}(V)$ where V is the set of voxels of I_D . A well-known classical result can immediately be derived:

Proposition 2 Let I_D a binary 3D image, being D the foreground set of voxels. The 6-Euler-Poincaré characteristic of I_D coincides with the Euler-Poincaré characteristic of the geometric ACC $Cell_6(D)$. The 26-Euler-Poincaré characteristic of I_D coincides with the Euler-Poincaré number of $Cell_{26}(D)$

It is clear that $G(Cell(I_D))$ is the connectivity graph for both $Cell_6(I_D)$ and $Cell_{26}(I_D)$. An (unoriented) edge c, c' of $G(Cell(I_D))$ with $dim_{26}(c) = dim_{26}(c') - 1$ (resp. of $Cell_6(I_D)$) generates a primal (resp. dual) vector (c, c') of $Cell_{26}(I_D)$ (resp. of $Cell_6(I_D)$). In this way,

both ACCs (and their associated DACCs) can be interpreted as symmetric systems, in the sense that if a cell c is connected to another c' in $G(\text{Cell}(I_D))$ by a directed path p (sequence of primal and dual vectors), then c' is also connected to c by a path having the same edges as p , but changing primal for dual vectors and vice versa.

In fact, the main DACCs of interest for the (6, 26)-HomTree representation are $\text{DACC}(\text{Cell}_6(D))$ and $\text{DACC}(\text{Cell}_{26}(I_D \setminus D))$.

3.2 Conceptual model

The graph-based HSF framework computes homological information of objects embedded in a digital image I_D , which are ACCs generated by maximally connected sets of voxels having the same color. This framework is mainly based on the construction of a smaller (exclusively in terms of vectors involved) asymmetric system (that is, a directed graph) within the connectivity graph of I_D , preserving cell path-connectivity [34]. Let us emphasize that although directed graphs are needed for the construction of an HSF, graphs belonging to an HSF representation are non-directed. From now on, we will refer to homological properties of voxel connected components, as to the properties of their associated ACCs.

The goal of this topological graph-based data structure is to save cell-nodes (given at inter-voxel level) and their incidence relationships within $G(\text{Cell}(I_D))$, in such a way that only graph transformations over the structure are needed to correctly and efficiently retrieve global topological information of both the foreground D and the background $I_D \setminus D$ (for example, connected components, region's Euler-Poincaré number, etc.). More precisely, a Homological Spanning Forest model (HSF, for short) on a (6, 26)-image I_D is a graded set of graphs with cell-nodes of dimension k and $(k + 1)$ ($k = 0, 1, 2$) such that the topological properties of I_D can be easily deduced from them.

Definition 4 [9] An HSF model of a geometric ACC $K = (K, B, \text{dim})$ of dimension n is formed by a set of connected incidence graphs involving k and $(k + 1)$ -cells, being $0 \leq k \leq n - 1$ and satisfying the following conditions:

- (a) HSF-graphs of the model involving 0 and 1-cells are trees with the maximal number of 0-cells connected through the minimum possible number of 1-cells. In other words, the number of 0-cell nodes in such a tree is equal to the number of its 1-cell nodes plus one. The set of such graphs is denoted by (0, 1)-HSF of K .
- (b) HSF graphs involving k and $(k + 1)$ -cells ($1 \leq k \leq n - 1$) are graphs (non necessarily trees) having the maximal number of k -cells (that are not involved in the $(k - 1, k)$ -HSF graphs of the model) connected through the minimum possible number of $(k + 1)$ -cells. The set of such graphs is denoted by $(k, k + 1)$ -HSF of K .

An HSF-model of an ACC is obviously a subgraph of its associated DACC. Here, we are mainly interested in a special kind of HSF models:

Definition 5 An (6, 26)-frontier-adapted HSF for a binary 3D image I_D is an HSF model of I_D , such that restricted to ACCs $\text{Cell}_6(D)$ and $\text{Cell}_{26}(I_D \setminus D)$ they are true HSF-models of them.

In light of this definition, an HSF model of the ACC $\text{Cell}_6(D)$ (respectively, $\text{Cell}_{26}(D)$) allows a straightforward computation of the 6-adjacency (resp. 26-adjacency) Euler-Poincaré characteristic of the image I_D . The main theoretical result of this paper can therefore be stated.

Theorem 2 Let I_D be a binary 3D image. Its (6, 26)-Hom-Tree model can be constructed based on the computation of any (6, 26)-frontier-adapted HSF model of I_D .

Sketch of the proof of Theorem 2:

We only give here a sketch of the proof, and we refer to Section 4 for design and implementation details of the different algorithms involved in the *Hom-Tree* construction. Here, we simply outline the main ideas that support the construction of the *Hom-Tree* structure from an HSF representation of the image, both globally and locally.

Global computation Given a binary image I_D with a set of black 6-CCs $\{R_1, \dots, R_b\}$ and a set of white 26-CCs $\{S_1, \dots, S_w\}$, the output of a frontier-adapted HSF construction algorithm gives, for each black or white connected component R , a set of graphs $HSF(R)$ that covers all cells of R at the inter-voxel level. Each graph of this kind has as nodes k -cells and $(k + 1)$ -cells ($k \geq 0$) and is denoted as the $(k, k + 1)$ -graph.

A $(k, k + 1)$ -graph G is essential if $\chi(G) = \# \{k\text{-cell nodes of } G\} - \# \{(k + 1)\text{-cell nodes of } G\} \neq 0$ (where $\#$ means cardinality of a set). Respectively, G is inessential when $\chi(G)$ is equal to 0. For each (black or white) connected component R of I_D , there is exactly one essential $(0, 1)$ - $HSF(R)$ tree (its 0-cells must be connected through a set of 1-cells since the object is connected) and the Euler-Poincaré characteristic $\chi(R)$ of R as cell complex (that is, $\chi(R) = \# \{0\text{-cells in } R\} - \# \{1\text{-cells in } R\} + \# \{2\text{-cells in } R\} - \# \{3\text{-cells in } R\}$) agrees with the alternate sum

$$1 - \sum_{G \in (1, 2)\text{-graph of } R} \chi(G) + \sum_{G \in (2, 3)\text{-graph of } R} \chi(G)$$

Taking advantage of the duality properties of the homological characteristics of the black 6-CCs and white 26-CCs in 3-dimensional ambience, we can deduce a new isotopic invariant of I_D , by adding weights to the edges of the classical rooted RAG tree of I_D . This tree corresponds to the $(6, 26)$ -*Hom-Tree*, and the weight of the edge (R_i, S_j) is given by the number of tunnels that the CCs R_i and S_j share [8].

Local computation Placing vectors (c, c') (where c and c' are a k -cell and a $(k + 1)$ -cell respectively) on the essential HSF-graph in a maximal way and satisfying that any cell belongs to at most one vector (see [35]), we are able to represent homological characteristics of I_D at connectivity graph level. Hence, we can define *critical cells* as those that remain unpaired within the essential HSF $(k, k + 1)$ -graphs, which must evidently have dimension k . Let us limit ourselves to say that the combinatorial homology features of any region R are intimately associated to the number of critical cells of its essential HSF sub-graphs on a frontier-adapted HSF. In particular, the number of tunnels of R agrees with the number of critical 1-cells existing in the $(1, 2)$ -HSF(R) sub-graphs. This particular region-growing strategy at inter-voxel level is sequentially guided by two criteria:

- (a) Merging through the boundary: A k -cell c (cell with dimension $k, k \geq 1$) and all the $(k - 1)$ -cells of $\partial\kappa(c)$ are included as nodes in a HSF-graph G of dimension $(k - 1, k)$ (composed by $(k - 1)$ -cells and k -cells) if there is an odd number of cells of $\partial\kappa(c)$ belonging to G .
- (b) Region color similarity: Any cell is endowed with one color and a color-dependant dimension. Therefore, region color similarity at the inter-voxel level is prioritized.

Having said that, the idea to construct a frontier-adapted HSF is based on the cellular technique of crack transport (see [18]). During the HSF construction process, the $(k, k + 1)$ -edges can be classified into primal and dual $(k, k + 1)$ -vectors, so that each HSF graph can

be interpreted as a directed graph. In this context, when we say that a cell falls into a critical one, we mean that they belong to the same essential HSF graph. A k -crack represents the boundary relations of a $(k + 1)$ -cell expressed under vector format. Given a k -cell c and a $(k + 1)$ -cell c' , with c being in the boundary of c' , a primal vector is a $(k, k + 1)$ -vector connecting the tail c with the head c' . The opposite edge, a $(k, k + 1)$ -vector connecting c' with c , is a dual vector. Thus, a k -crack contains exactly one primal vector (c, c') and as many dual vectors (c', c'') as k -cells are in the boundary of c' , being $c'' \neq c$.

In order to correctly adapt the HSF computation algorithm to the computation of the *Hom-Tree* (see step (d) in Fig. 3), we first need to work on the auto-dual cubical grid, specifying the contribution of each black and white set C of eight mutually 6-adjacent voxels to the global computation of an HSF of I_D and of its corresponding CCs. For instance, if we have a set C of this type with only one white voxel, its contribution in terms of cells to a white 26-CC is of: one 0-cells, six 1-cells, eleven 2-cells and seven 3-cells; and its contribution in terms of cells to a black 6-CC is reduced to one 0-cell. Therefore, as mentioned before, each inter-voxel element of the auto-dual cubical grid is endowed with a unique color and with a color-dependent dimension, before constructing an HSF of I_D .

The rest of steps in Fig. 3 are to be detailed in Section 4.

Let us note that the cells of an inter-voxel frontier cubical complex between two regions R_i and S_j shape a connected boundary surface that is the geometric intersection of the set of physical voxels of R_i with the ones of S_j . Finally, boundary and coboundary operations need to be applied to the critical cells of the different black 6-CCs and white 26-CC, so that the *Hom-Tree* representation is obtained. Firstly, black critical 0-cells (CCs) are paired by duality to white critical 2-cells (cavities) and vice-versa; then critical 1-cells of black 6-CC (tunnels) are paired by duality with critical 1 cells of its neighboring white 26-CC. After the first step, the edges of the RAG tree of I_D are determined. After the second one, the weights of the edges are computed. Note that it is therefore correct to speak of *tunnels shared by a black 6-CC and a white 26-CC*. In fact, these tunnels are shared by their associated image ACCs.

A straightforward consequence of Theorem 2 is the following one:

Corollary 1 *There is an algorithm for computing the 6-adjacency (resp. 26-adjacency) Euler-Poincaré number of I_D based on the computation of any (6, 26)-frontier-adapted HSF model of I_D .*

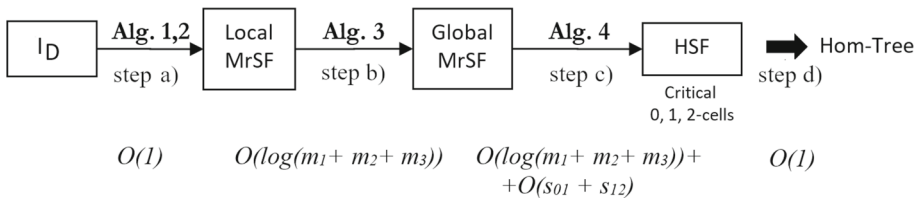


Fig. 3 Main steps of the *Hom-Tree* computation via HSF. The computation of a frontier-adapted HSF is the last step performed by Algorithm 4. The *Hom-Tree* can be straightforwardly extracted once the HSF structure has been obtained (see Section 4)

4 Parallel generation of Hom-Tree

The main steps of the complete construction process of Theorem 2 are shown in Fig. 3. The proposed algorithm is divided into four steps: (a) Local MrSF (Morse Spanning Forest) construction; (b) Global MrSF construction; (c) Frontier-adapted HSF; and (d) Hom-Tree generation. These steps are detailed in the following subsections. Due to the fact that implementing the general case would imply a more complex treatment and development, we restrict ourselves here to detail the computation of the (6, 26)-Hom-Tree for well-composed images (in which every 26-CC of constant color is also a 6-CC). Note that in this case of well-composedness, the (6, 26)-Hom-Tree is isomorphic to the (26, 6)-Hom-Tree.

The complexity order of the complete process remains close to the logarithm of the sum of the image’s dimensions. More concretely, step (a) is fully parallel, that is, $O(1)$, step (b) has a maximum time order of $\log_2(m_1 + m_2 + m_3)$, step (c) also has logarithmic time, and step (d) is fully parallel. Figure 3 summarizes the time orders for the different steps of the whole HSF calculation considering $m_1 \times m_2 \times m_3$ computation processors. Pseudocodes explaining each step have been formally written (see Algorithms 1 and 2 for step (a), Algorithm 3 for step (b) and Algorithm 4 for step (c)). Note that these pseudocodes describe the sequential versions of the corresponding parallel algorithms involved within the Hom-Tree computation. The MATLAB/OCTAVE implementation presented here faithfully follows these sequential pseudocodes. The theoretical complexity of the parallel algorithms is as well detailed. Note that the algorithm pseudocodes only show those essential matrices for HSF computations. Additional matrices used for codification are defined in the MATLAB/OCTAVE codes.

With the aim of clearly showing the algorithm’s output in every step, a simple GUI (Graphical User Interface) has been as well implemented (see Fig. 4). This software has been used to produce the figures of this Section.



Fig. 4 GUI of the Hom-Tree application

4.1 MrSF construction at local level: activation of processing units

Taking as embedding scaffolding of a given image I_D its associated DACC $\text{Cell}_6(I_D)$, the 0-cells inherit the properties (such as color; black or white from now on) of the voxels of I_D . Within our parallel framework, one processing element is defined per voxel. Given a foreground voxel v , each processing unit $PU(v)$ is centred in the common corner of the eight mutually 26-adjacent voxels, where v corresponds to the voxel located in the lower left front corner (Fig. 5).

Algorithm 1 returns two matrices that are directly obtained from the DACC grid. The first one contains the color assigned to each k -dimensional cell according to its neighbors. The second one stores the set of potential critical cells. During the construction of such matrices, and with the aim of promoting parallelism, for each voxel, we only check its 6-adjacent voxels towards certain predefined directions (+Z,+Y,+X in our implementation).

For example, in the case of foreground voxels, 0-cells are considered as potentially critical if there is no other foreground voxel towards directions (+Z,+Y,+X) (intuitively, this means that these background adjacent 0-cells form a corner). In the general case, a k -cell is potentially critical if at least one of its coboundary $(k + 1)$ -cells towards directions (+Z,+Y,+X) has a different color than the k -cell. As a consequence, in Fig. 7 a foreground 2-cell is selected as potentially critical if it is located in the vertex of a corner that is parallel to the first octant, and potential foreground critical 1-cells appear for L-shapes, oriented towards the positive axes. 3-cells are never critical for 3D images. To sum up, potential critical cells at this stage are those having cells of different color on their borders.

After Algorithm 1, an initial HSF of the whole image, called Morse Spanning Forest $MrSF(I_D)$, is built (see Algorithm 2). It is necessary to emphasize that the resulting MrSF graphs may not necessarily be trees and will then be pruned to trees, so that parallel computing is promoted. Roughly speaking, an $MrSF(I_D)$ is composed of a set of $(k, k + 1)$ -trees ($k = 0, 1, 2$), having as nodes cells of dimension k and cells of dimension $k + 1$, and having as edges some incidence relations between these cells, called $(k, k + 1)$ -edges (see Definition 4).

In Fig. 5 (left), a visual example of all the cells and trees of a given MrSF are depicted. This figure represents a ring parallel to the XOY plane. The circles, triangles, squares and

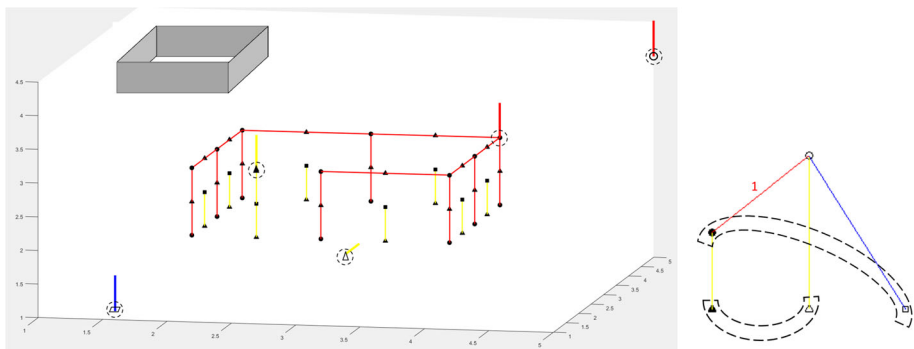


Fig. 5 Left: A gross ring perpendicular to Z axis. For clarity purpose, only foreground trees are depicted. Critical cells are highlighted by surrounding them with a dashed circle. Right: It is Hom-Tree. Solid symbols are for foreground cells and hollow symbols for the background ones. Associations among these cells are shown with dashed arcs. The topological duality properties from which these associations are extracted can be found in Section 4.4

Algorithm 1 Cell values and possible critical cells

Preliminary definitions for 3D images:

$DIM=3$; canvas dimension
 $n_c(k) = (1,3,3,1)$ is the number of k -cells within a voxel; $k = 0,1,2,3$;
 $n_{adj_c}(k) = (1,2,4,8)$ is the number of adjacent voxels of a k -cell.
 // Each voxel v in I is defined by its integer coordinates. Voxel values are: background = 1; foreground=0.
 // Each cell is defined by $(k,c(k),v)$ with $c(k)=1, \dots, n_c(k)$. In addition, each cell can have up to $2 * DIM$ possible directions for vectors coming out of it.

Input:

3D Image I of dimensions $m_1 * m_2 * m_3$

Output:

cell_value: cell_value of 0-cells is simply the voxel color. For the rest of cells, it represents the sum of background voxels that a k -cell is surrounded by.

possible_crit_cells: it indicates if a cell is critical. During the transport stage some false critical cells are to be cancelled.

for each voxel v in I **do**

 // v is defined by its integer coordinates

for $k = 0,1,2,3$ **do**

for $c = 1, \dots, n_c(k)$ **do**

$cell_value(k,c,v) = \sum\{adj_{0cells}(d,c,v)\}$

 // $\{adj_{0cells}(k,c,v)\}$ is the set of $n_{adj_c}(k)$ voxels (0-cells in this convention) adjacent to a k -cell

$possible_crit_cells(k,c,v) = f_crit_cell(k,c,dir,act_state(v))$;

 // f_crit_cell returns a Boolean value indicating if the cell may be a critical one.

end for

end for

end for

// theoretical time complexity is simply $O(1)$ due that there are no loop-carried dependences among iterations; thus all the iterations can be done in parallel.

stars represent 0-cells, 1-cells, 2-cells and 3-cells, respectively. Foreground cells are drawn with solid shapes, and hollow shapes are used for background cells. There are (0, 1), (1, 2) and (2, 3)-trees, represented by red, yellow, and blue lines, respectively. The thickest lines represent the critical cells of the corresponding tree.

Given a three-dimensional image, the construction of the MrSF lies within the fact that there are nine possible activation states for any processing element, which are depicted in Fig. 6. Note that, as mentioned above, one processing element per voxel is considered. It should be understood that when we say that a vector belongs to an HSF tree, it means that its corresponding edge (in terms of connection), tail, and head (in terms of nodes) are included in such a tree.

A single primal (0, 1)-vector v is considered to be included within the (0, 1)-tree of each processing element (marked in red in Fig. 5). The corresponding dual (1, 0)-vector (whose tail is the head of v) is likewise included within the same tree. The rules followed to build the (1, 2) and (2, 3)-trees are similar. In these cases, up to two dual vectors may go from a $(k + 1)$ -cell to a k -cell ($k = 1, 2$). An obvious rule for the preservation of the trees must be maintained at this stage: Only those dual vectors that connect the same dimensional tree are allowed. Note that dual inter-voxel connections can be extensive, that is, we can activate all the possible dual vectors that do not contravene the tree allocation for each cell. However, as we are interested in building trees (but not graphs) for efficient parallel processing of crack transports (see Section 4.3) some dual (2, 1)-vectors are not activated in our current implementation. More exactly, for those k -cells that can receive two dual vectors coming from $(k + 1)$ -cells ($k = 1, 2$), only one of them is activated.

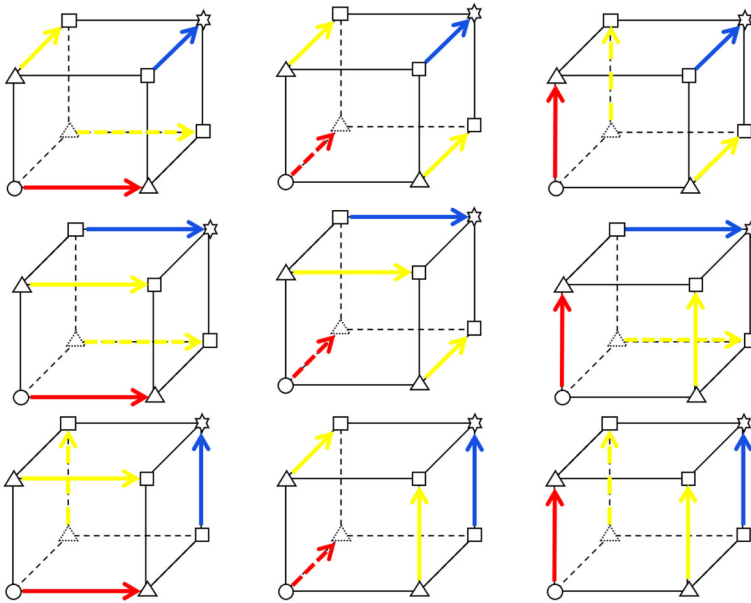


Fig. 6 Activation states for a processing unit with its primal vectors. Each processing unit contains eight active cells: one 0-cell (drawn with an empty circle), three 1-cells (triangles), three 2-cells (squares) and one 3-cell (star). The red, yellow and blue arrows belong to the (0, 1), (1, 2), (2, 3)-trees, resp. Left: Three states when the (0, 1) primal vector goes to +X. Center: Three state when the (0, 1) primal vector goes to +Y. Right: Three states when the (0, 1) primal vector goes to +Z

For some objects, the set of possible critical cells obtained through these local interactions is the correct one. This is the case of the object of Fig. 5 (left). Critical cells are drawn by our tool with thicker lines: foreground contains one critical 0-cell and one critical 1-cell, and background contains one critical 0-cell, one critical 1-cell and one critical 2-cell. The main advantage of this local MrSF construction method is that it can be computed in a fully parallel manner for each voxel (timing order is $O(1)$). The topological duality properties that allow us to obtain the relations between critical cells that are depicted in Fig. 5 (right) are explained in Section 4.4.

4.2 Global MrSF construction

Once a (local) MrSF has been built, its trees are to be labeled in terms of connected graphs (see Algorithm 3), obtaining what we call the global MrSF.

This labeling consists of tracking the label from every cell to a new one. Note that potential critical cells are labeled with a unique identifier and are the roots of a tree (thus, they sink to themselves). There may also exist cells that are the roots of inessential trees. Additionally, jump distances from any cell to the critical cell of its belonging tree can be easily computed here. Most existing labeling algorithms perform this tracking using the label number stored at each site ([36]). That is, they compute $label[i] = label[label[i]]$, where the element $label[i]$ works as an index to search for a different element in the same list $label[i]$, until the label remains unchanged. We refer the reader to Section 3 and Algorithm 1 of [36] for a detailed description of these types of algorithmic processes. In our case, we use two additional pieces

Algorithm 2 Local MrSF (MrSF construction at local level: Activation of processing units)

```

// see Preliminary definitions for 3D images in Algorithm 1
Input: 3D Image  $I$ ,  $cell\_value$  (defined in Algorithm 1)
Output:

$pACC$ vectors: it contains for each cell and for each  $2 * DIM$  directions a Boolean value indicating if this vector exists. Cracks can be easily obtained from  $pACC$ vectors.

for each voxel  $v$  in  $I$  do
    //  $v$  is defined by its integer coordinates
     $act\_state(v) = f\_act\_state(cell\_value(k,c,v))$ ;  $k = 1,2,3$ ;  $c = 1, \dots, n_c(k)$ 
    //  $f\_act\_state$  defines the nine possible activation states for primal vectors in a voxel, which result from: its 0-cell can have three directions (3 cases) and, independently, for each of these directions, the three 2-cells can be paired with the 3-cell. The remaining two 1-cells and two 2-cells must be linked in a unique way.
    // Preference rules for the vector directions are set in this function.
    // now primal vectors can be marked.
    for  $k = 0,1,2$  do
        for  $c = 1, \dots, n_c(k)$  do
            for  $dir = 1, \dots, 2 * DIM$  do
                 $pACC$ vectors( $k,c,dir,v$ ) =  $f\_primal(k,c,dir,act\_state(v))$ ;
                //  $f\_primal$  defines returns a Boolean value indicating if the vector is activated.
                 $cell\_membership(k,c,v) = f\_membership(k,c,act\_state(v))$ ;
                // 0 and 3 cells always belong to (0,1) and (2,3) trees resp. However, 1-cell membership can be (0,1) or (1,2) trees. Likewise, 2-cell membership can be (2,3) or (1,2) trees. These memberships are booked here.
            end for
        end for
    end for
    // those vectors that were not booked as primal may be marked as dual.
    for  $k = 1,2,3$  do
        for  $c = 1, \dots, n_c(k)$  do
            for  $dir = 1, \dots, 2 * DIM$  do
                if ( $pACC$ vectors( $k,c,dir,v$ ) == 0) then
                     $pACC$ vectors( $k,c,dir,v$ ) =  $f\_dual(k,c,dir,cell\_membership(neigh(k,c,dir,v)),act\_state(v))$ ;
                    //  $f\_dual$  determines if a dual vector must be set. Pruning of some (1,2) vectors are done in this stage. This pruning implies that some paths along the whole (1,2) tree are lost.
                    //  $neigh(k,c,dir,v)$  returns the neighbor cell coordinates to that cell given by ( $k,c,v$ ) in the direction  $dir$ .
                end if
            end for
        end for
    end for
    // theoretical time complexity is simply  $O(1)$  due that there are no loop-carried dependences among iterations; thus all the iterations can be done in parallel.

```

of information: which of the cells may be critical (whose labels will remain unchanged) and the magnitudes and directions of the hops from one cell to the root. Note that the first jump is immediately given by the local MrSF information. To build trees at this stage, additional prunings within the resulting graphs might be done for the (1)-cracks.

According to the definition of primal and dual vectors from the previous stages, it is obvious that the maximum distance between two cells is $(m_1 + m_2 + m_3)$. Besides, using the process $label[i] = label[label[i]]$, the jump distances increase in a geometric manner of ratio 2.

4.3 Frontier-adapted HSF construction via crack transports

The final step consists of minimizing the number of critical cells, leading to a new HSF that is frontier-adapted (see Algorithm 4). From now on, this final HSF will be named $HSF(I_D)$.

Algorithm 3 Global MrSF

```

// see Preliminary definitions for 3D images in Algorithm 1
Input:
3D Image  $I$ ,  $pACC$  vectors (defined in Algorithm 2),  $cell\_value$  (defined in Algorithm 1)
Output:
 $cell\_labels$ : a label indicating the root cell that a cell sinks to. Critical cells are labelled with a unique identifier and are the
roots of a tree (thus, they sink to themselves). There are other cells that are root nodes of inessential trees.
// Additionally, jump distances from any cell to the critical cell of the tree it belongs to can be easily computed here.

// initial values of  $cell\_labels$ 
for  $k = 0, 1, 2, 3$  do
  for each voxel  $v$  in  $I$  do //  $v$  is defined by its integer coordinates
    for  $c = 1, \dots, n_c(k)$  do
       $cell\_labels(k, c, v) = f\_initial\_label(k, c, v)$ ;
      //  $f\_initial\_label(cell\_labels(k, c, v))$  returns for a cell a label (linear direction in MATLAB/OCTAVE
      notation) if the cell is critical, or the linear direction of the nearest cell of its same dimension. Here "nearest"
      is computed following the tree that the cell belongs to.
    end for
  end for
end for

// Global MrSF building
 $max\_hop\_links = \log_2(m_1 + m_2 + m_3 - 1) + 1$ ;
for  $k = 0, 1, 2, 3$  do
  for  $nof\_hops = 1, \dots, max\_hop\_links$  do
    for each voxel  $v$  in  $I$  do //  $v$  is defined by its integer coordinates
      for  $c = 1, \dots, n_c(k)$  do
         $cell\_labels(k, c, v) = cell\_labels(cell\_labels(k, c, v))$ ;
      end for
    end for
  end for
end for
// theoretical time complexity of the first loop for  $k = 0, 1, 2, 3$  do is simply  $O(1)$  due that there are no loop-carried
dependences among iterations; thus all the iterations can be done in parallel.
// theoretical time complexity of the second loop for  $k = 0, 1, 2, 3$  do is the logarithm of  $max\_hop\_links$ , that is,  $O(\log_2$ 
 $(m_1 + m_2 + m_3))$ , because each iteration of the loop for  $nof\_hops$  must be done in order. Similarly to Algorithms 1 and 2,
the rest of loops have no loop-carried dependences.

```

Within this HSF, taking into consideration only cells belonging to the 6-cellularization of the foreground, we have an HSF of D (in fact, of $Cell_6(D)$); however, if considering only cells belonging to the 26-cellularization of the background, we have the HSF of its complement (that is, of $Cell_{26}(I_D \setminus D)$).

As we have commented before, in some occasions, the MrSF is already a frontier-adapted HSF. This is because every k -cell can be paired to a $(k - 1)$ -cell towards a positive direction (+X, +Y or +Z). In these cases, no crack transports are needed. This is the case of Fig. 5 (left). Only the 0 cell with the highest coordinate values, that is $(X, Y, Z) = (4, 4, 3)$, cannot be paired to a 1-cell. Likewise, the 1-cell located at $(X, Y, Z) = (1.5, 2, 3)$, cannot be matched to a foreground 2-cell, being therefore the critical cell of the ring tunnel. Proceeding in a similar way for the background, a 2-cell $(X, Y, Z) = (1.5, 1, 1)$ represents its cavity and a 1-cell $(X, Y, Z) = (2.5, 0.5, 1.5)$ is the critical cell of its tunnel.

The general case implies a lot of crack transports in order to minimize the number of critical cells of an object. In most of the cases, many transports can be performed in parallel. Bearing in mind 6-adjacency and in order to promote parallelism, cracks are only defined toward positive directions: +X, +Y, +Z. These cracks will allow one to build the different trees for a given image. The simplest way to ensure that a pair of redundant critical k and

Algorithm 4 Frontier-adapted HSF construction via crack transports

```

// see Preliminary definitions for 3D images in Algorithm 1
Input: 3D Image  $I$ ,  $pACC$  vectors (defined in Algorithm 2),  $cell\_value$  and  $possible\_crit\_cells$  (defined in Algorithm 1),
 $cell\_labels$  (defined in Algorithm 3)
Output:
 $cell\_labels$ : see Algorithm 3. In truth, this matrix represents the three HSF trees, that is, the (0-1), (1-2), (2-3) trees.
 $final\_crit\_cells$ : it contains the final critical cells.
// Additionally, tree bounding relations can be computed in a similar way to the Algorithm 3, using the final  $cell\_labels$ .
// Parallel 0/1 pairs Cancellation
 $n_{01canceled\_pairs} = 0$ ;
do
  for each critical 0-cell  $cc0$  in  $possible\_crit\_cells$  do
    for  $traveling\_dir = 3, 2, 1$  do
       $cc_{neigh} = neigh\_cell(cc0, traveling\_dir)$ ;
      //  $neigh\_cell()$  returns the nearest 1-cell towards the direction  $traveling\_dir$ . See Appendix A.
      // first 2-cell (co-border of the 1-cell towards a negative direction: -X, -Y or -Z)
       $cc_{start} = next\_cell(cc_{neigh}, cell\_labels)$ ;
      //  $next\_cell()$  returns the nearest 2-cell of the  $cc_{neigh}$  in the (1,2) tree (Fig. 1 of the Appendix A)
       $cc_{end} = cell\_labels(cc_{start})$ ;
      //  $cc_{end}$  is the parent node (some 2-cell) in the MrSF
       $cc1 = neigh\_cell(cc_{end}, pACC$  vectors( $cc_{end}$ ));
      //  $cc_{end}$  points now to a 1-cell  $cc1$  which has two boundary 0-cells, called here  $cc_{start\_back}(1;2)$ . In order to uncover a 0/1 pair
      // cancellation, we check if any of these  $cc_{start\_back}$  goes backwards to the beginning  $cc0$  along the (0,1) tree (see Appendix A).
      for  $k = 1, 2$  do
         $cc_{end\_back} = cell\_labels(cc_{start\_back}(k))$ ;
        if ( $cc_{end\_back} == cc0$ ) then
          // if  $cc_{end\_back}$  were the same as  $cc0$ , the crack transport would be done, the MrSF (represented by  $cell\_labels$ ) would be
          // appropriately changed via function  $doing\_transport()$ , and the pair of false critical cells would be deleted.
          // Note that concurrency among different 0/1 pair cancellations is guaranteed if these two travels from  $cc0$  to
          //  $cc_{start\_back}(k)$  and for  $cc_{start\_back}(k)$  to  $cc_{end\_back}$  form a cycle (because MrSF is only composed of trees).
           $cell\_labels = doing\_transport(cell\_labels, cc0, cc1)$ ; //
           $possible\_crit\_cells(cc0) = 0$ ;
           $possible\_crit\_cells(cc1) = 0$ ;
           $n_{01canceled\_pairs} = n_{01canceled\_pairs} + 1$ ;
        end if
      end for
      // similar operations to that of the first 2-cell for the second 2-cell (the other 2-cell co-border of the 1-cell towards
      // negative direction given by  $cc_{neigh}$ ; see Appendix A). Additional 0/1 pair cancellations may also occur here.
    end for
  end for
while ( $n_{01canceled\_pairs} != 0$ );

// some 0/1 pair cancellations may not be detected using previous operations, because of the pruning done in the (1-2)
// trees. Thus, the last 0/1 pair cancellations must be done sequentially by the following checking (done from the remaining
// possible critical 1-cells towards the possible critical 0-cells.
for each critical 1-cell  $cc1$  in  $possible\_crit\_cells$  do
  // Checking if the two boundary 0-cells (called "negative" and "positive" here) sinks towards different 0-cells.
   $cc_{positive} = cell\_labels(neigh\_cell(cc1, +1))$ ;
   $cc_{negative} = cell\_labels(neigh\_cell(cc1, -1))$ ;
  if ( $cc_{positive} != cc_{negative}$ ) then
    // if  $cc_{positive}$  and  $cc_{negative}$  were different, the crack transport would be done, the MrSF (represented by  $cell\_labels$ )
    // would be appropriately changed via function  $doing\_transport()$ , and the pair of false critical cells would be deleted.
    // Any of both  $cc_{positive}$  or  $cc_{negative}$  can be chosen for the cancellation.
     $cell\_labels = doing\_transport(cell\_labels, cc_{positive}, cc1)$ ;
     $possible\_crit\_cells(cc0) = 0$ ;
     $possible\_crit\_cells(cc1) = 0$ ;
     $n_{01canceled\_pairs} = n_{01canceled\_pairs} + 1$ ;
    // A total of  $s_{01}$  sequential cancellations of remaining 0/1 cells resp. are done here.
  end if
end for
// Similar operations for finding 1/2 pair cancellations are done subsequently. A total of  $s_{12}$  sequential cancellations of
// remaining 1/2 cells resp. are done after the parallel 1/2 pair cancellations.
 $final\_crit\_cells = possible\_crit\_cells$ ;

// theoretical time complexity of Algorithm 4 depends on its two parts. For part 1, it is the number of iterations of the loop
// do...while (), which depends on the condition ( $n_{01canceled\_pairs} != 0$ ). Like Algorithms 1 and 2, its inner loops have no loop-
// carried dependences. For part 2, theoretical time complexity of the last for loop is proportional to its number of
// sequential cancellations  $s_{01} + s_{12}$ . This is the only completely sequential part (see text for more details).

```

$(k + 1)$ -cells (referred as crt and crt' respectively) can be transported in parallel (with other pairs) is by first inspecting the $(k + 2, k + 1)$ -tree belonging to the coboundary of crt (represented as dashed black lines in Fig. 7, bottom left), and then the $(k, k + 1)$ -tree that goes in the reverse direction (represented as dashed red lines in the same Fig.). Note that the $(k + 2, k + 1)$ -tree was built from $(k + 2)$ -cells to $(k + 1)$ -cells to promote parallel transports; so it is considered as a directed one. In this figure, dots, triangles and diamonds represent k , $(k + 1)$ and $(k + 2)$ -cells, respectively. When the reverse path from crt falls again in crt' , this pair of paths generates a cycle, and the transport of crt and crt' can be done in parallel (with many other transports, which would produce others different cycles). This would happen in Fig. 7 (top left) for the two pairs of 0 and 1 cells. Once a redundant pair is detected, the crack transport proceeds as follows: Firstly, the edge that connects crt to a $(k + 1)$ -cell of a different object is deleted (represented by an arrow over critical cells in Fig. 7, top and bottom left). All edges that connect crt' with the $(k + 1, k + 2)$ -tree must as well be erased, because crt' now belongs to a $(k, k + 1)$ -tree. After that, these cells are no longer critical. Secondly, new edges (solid black lines in Fig. 7, bottom left) need to be added to connect crt' to some of its boundary cells. The boundary cells to be connected are: those belonging to the tree of the no longer critical k -cell crt (left dot) and those belonging to the tree of the remaining critical k -cell of the same object (right dot). After that, the $(k + 1)$ -cell belongs to a $(k, k + 1)$ -tree. This parallel transport process corresponds to the Algorithm 4). In the next paragraph, parallel transports are explained using Fig. 7 for the 0, 1 and 2 (false) critical cells. Note that the first loop in Algorithm 4 ends when no cancelation of false critical cells can still be done (variable $n_{01cancelled_pairs}$ remains to be null). Then, the rest of false critical cell

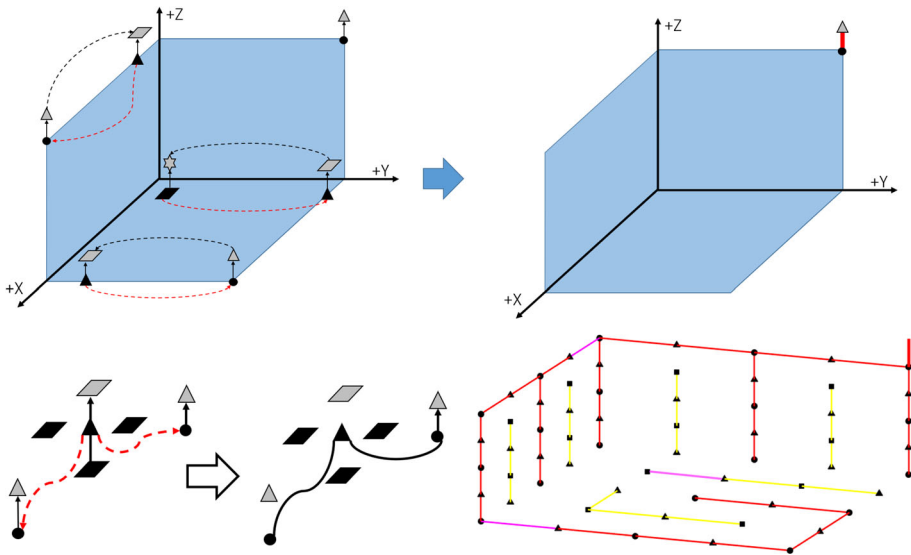


Fig. 7 Crack transports. Top Left: An object composed by three planes parallel to the axis yields initially to three critical 0-cells, three critical 1-cells and one critical 2-cell that appear in the MrSF. Arrows are the primal vectors to their nearest contour cells. The arrows indicate the forward and backward paths that have to be followed to pair the false critical cells. Top Right: resultant critical 0-cell of the HSF after doing the transports. Bottom Left: The detection of a pair of redundant k and $k + 1$ critical cells and the crack transport process. Here, dots, triangles and diamonds are k , $k + 1$ and $k + 2$ cells, resp. Arrows are also the primal vectors to their nearest contour cells, dashed red lines express trees. Bottom Right: HSF returned by our tool showing only one critical foreground 0-cell, marked with a thicker red segment. Pink segments are edges coming from the transports

pairs are to be sequentially canceled in the last *for* loop, so that at the end of this algorithm we always get a set of final critical cells. Finally if some transports could not be carried out in parallel, we would proceed as explained in [9].

Our tool converts the MrSF into the HSF by transporting in a first step those false critical 0-cells with false critical 1-cells, using the (1, 2)-tree for the forward path and the (0, 1) for the reverse path. When the reverse path falls again in the initial critical 0-cell, the transport can be done in parallel with many other transports, as it supposes a unique path along a pair of trees. In the case of Fig. 7, two pairs of false critical 0 and 1-cells are canceled. The last critical 0-cell (upper and most right cell in the OYZ plane) cannot be canceled, as it remains as the representative of the object (one connected component). In a second step, those false critical 1-cells are canceled with critical 2-cells, using the (2, 3)-tree for the forward path and the (1, 2) for the reverse path. Once again, when the reverse path falls again in the initial critical 1-cell, the transport can be carried out in parallel with many other transports. For the object in Fig. 7, one pair of false critical 1 and 2-cells are canceled. Thus, no cavity nor tunnel remain for this object, as expected. As a result, the HSF must only contain one critical 0-cell after doing the transports (Fig. 7, right). The HSF returned by our tool is shown in Fig. 7, bottom, containing only one critical foreground 0-cell, marked with a thicker red segment. Pink segments are edges that come from the three transports explained previously.

The most basic tree representation consists of simply bookkeeping a label for each cell; thus, the crack transport can be reduced to a pointer-like switching in the label structures. The construction of the initial MrSF components and trees allows to transform an $MrSF(I_D)$ into an $HSF(D)$, via crack transports and achieving an elevated parallelism degree.

A complete implementation has been done and can be downloaded in [37]. Figure 11 of the Appendix A includes detailed information about this process.

The transport process may require several steps, so an intermediate MrSF may be produced at the end of each step until the HSF is achieved. In each step, a set of false critical 1-cells can be deleted (crack transports are performed) with the same number of false critical 0-cells. Similar operations can delete pairs of false critical 1-cells and 2-cells. In the *do...while* loop of the Algorithm 4, note that crack transports are performed only if cc_{end_back} are the same cells as cc_0 . In this case, the MrSF (represented by *cell_labels*) is appropriately changed through the function *doing_transport()*, and therefore the pair of false critical cells is deleted. Here we limit ourselves to highlighting that moving along trees guarantees that the uniqueness in the selection of the different couples of (0, 1)-cells (or (1, 2)-cells) is ensured, and therefore there is no need for synchronization primitives. In this sense, perfect parallelism is achieved in these transports; thus, their time order can be considered as $O(1)$. Similarly, for the pairing of false (1, 2)-MrSF critical cells, a similar parallel procedure can be performed, preserving the time order as $O(1)$. The number of incident cells is lower in this ulterior pairing. However, the transports along these paths for couples of (0, 1)-cells and (1, 2)-cells do not imply that complete pairing has been done and that these transports would give us the final HSF. There are complex shapes that need some subsequent iterations. The exact form of these shapes will be studied elsewhere; for the present work, the experimental results in the next section reveal that the number q of subsequent iterations needed for different types of images is less than 5 for most of the analyzed 3D images. Besides, some of the images tested required a sequential stage of (0, 1)-cell coupling that cannot be processed in parallel. This is due to the necessary pruning of some (1, 2)-vectors done in the previous stage. This pruning implies that some paths along the whole (1, 2)-tree are lost. As a consequence, some false 0-cells are not able to find the corresponding false critical 1-cell on some occasions. These (0, 1)-pair cancelations are described in the last part of Algorithm 4 (after the *do...while* loop). Note that if $cc_{positive}$ and $cc_{negative}$ are different, the crack transport is performed, MrSF (represented

by *cell_labels*) is then appropriately changed via the function *doing_transport()*, and the pair of false critical cells is therefore deleted. Either *cc_{positive}* or *cc_{negative}* can be chosen for cancellation.

As illustrated in the next section, the number of sequential transports (denoted s_{01} in Fig. 3) represents less than 0.5% of the number of voxels even for random images, which is around ten times lower for the natural images analyzed. Pruning was also done for the inverse path $2/1/2$, which means that some $(1, 2)$ -transports can either be done in parallel (which number is denoted as s_{12} in Fig. 3). Nevertheless, for real or random 3D images, the percentage of these sequential $(1, 2)$ -transports is negligible. In summary, the time order of one parallel step in stage (c) is $O(1)$. Besides, a small number q of parallel transport iterations is required for most of the images. Finally, the time order would diverge from $O(1)$ only for the small percentage of sequential transports. One final remark that highlights the practical significance of our method is that many useful geometric properties (area, perimeter, etc.) can be computed at the same time and also in parallel along with the HSF.

4.4 Hom-Tree construction

Given an *HSF*(D), we can directly obtain the *Hom-Tree* structure of I_D . Taking into account the fact that a tree is homotopy equivalent to a point, the $(0, 1)$ -trees of the different black and white CCs of I_D can be contracted within the $(0, 1)$ -tree of the whole image. According to the generation of an HSF structure, each critical 0-cell of a given color has a crack associated with it that connects it with the opposite color. Thus, the nesting of the component (and then the classical RAG tree) can be easily found. Moreover, those critical cells of higher dimensions can be associated with the component of their border cells. In 3D, tunnels add valuable information when relating each tunnel with a pair of foreground and background components. In addition, cavities (represented by critical 2-cells) have a dual relation with critical 0-cells. From now on, these relations are expressed in the *Hom-Trees* of the processed images, using dashed arcs to enclose related cells.

For example in Fig. 5 (right) the *Hom-Tree* for a gross ring of Fig. 5 (left). Solid symbols are used for foreground cells and hollow symbols for the background ones. The topological duality property existing between a foreground object and its surrounding background, is reflected within the HSF structure as follows: the foreground object as a CC (represented by a 0-cell) produces a cavity in the background (represented by a 2-cell); the foreground tunnel (represented by a 1-cell) induces another background tunnel (represented by a background 1-cell). These associations are shown with dashed arcs. Only the background critical 0-cell of the canvas, which is the root of the tree, does not present any association. Additionally, color associations are shown with yellow lines (both for background and for foreground cells). Apart from them, there is a red line, which corresponds to the fact that the foreground object is embedded in the background. This line is weighted with 1 because only a pair of tunnels appear.

Another interesting example where *Hom-Tree* can help distinguish it from other homologically equivalent (but not isotopically) objects (see Fig. 1) is the case of an empty cube with one diameter and an outer handle. Figure 8 shows the $(0, 1)$ -tree and all trees for the cells composing the foreground object. Note that our current implementation focuses on distinguishing non isotopically equal objects through the use of homological relations. A step further that is intended for future work is the generation of sets of 1-cycles that delineate homology groups with a minimal total length (as addressed from another point of view in [38]).

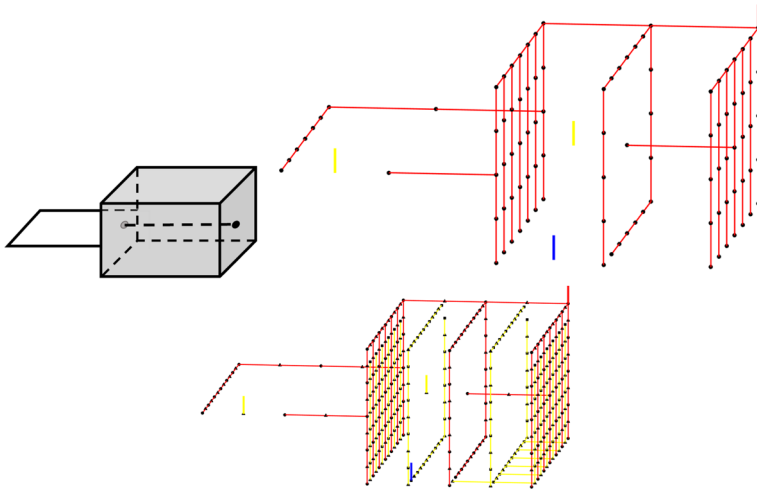


Fig. 8 Left: An empty cube with one diameter and an outer handle. Right: The $(0, 1)$ -tree of the cells composing the foreground object. Bottom: All the trees for the foreground object

Similarly in Fig. 9 (Top) these associations are depicted for a more complex image: A sphere with one diameter and an outer handle (composing the foreground object) that is surrounded by the background (see Fig. 9, Bottom). Again, the topological duality property of an image can be rediscovered within the HSF structure by pairing every critical foreground cell with its corresponding background counterpart (shown with dashed arcs). Foreground critical cells (solid symbols) are paired with the critical cells in the background (hollow symbols). In this example, the foreground CC is related to the background cavity, the foreground cavity of the sphere is paired to a background CC, and foreground tunnels are paired to background tunnels. Again, only one critical background 0-cell (the root of the canvas tree) does not present any association.

To sum up, once the 1 and 2 related cells have been identified, *Hom-Tree* can be simply reduced to the edges that connect the 0-cells (black and white little circles in Fig. 5, Right) plus a weight indicating the number of tunnels that hang from each component.

5 Experimental results and potential applications

A complete implementation of the proposed algorithm for the $(6, 6)$ -adjacency pair has been performed in the MATLAB / OCTAVE language ([37]). This pair is chosen for implementation to simplify the resulting code. Thus, every processing unit (for both background and foreground voxels) follows the foreground criterion explained in the previous section. The mentioned language favors an inherent parallel codifying, and special care has been taken to promote element-by-element matrix operations whenever possible. These operations can be executed theoretically in a fully parallel manner. Only a few parts of the whole process cannot be executed using this kind of operations, thus yielding a few loop structures including loop-carried dependencies. In order to test the correctness of Alg. 1, 2, 3, 4 and to evaluate their theoretical time complexity (under the assumption that a processing element is available for each cubical voxel), we proceed with a set of tests that measure the number of parallel

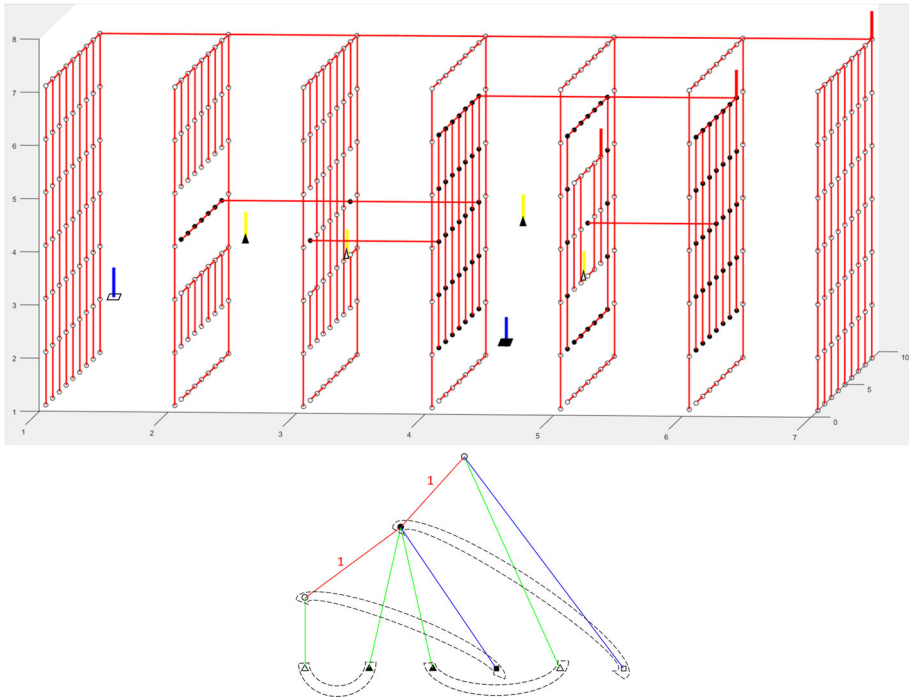


Fig. 9 Top: Same object as Fig. 8 (composing the foreground object) that is surrounded by the background. For clarity purpose, only the (0, 1)-tree is depicted. Bottom: It is *Hom-Tree*. Foreground cells are drawn with solid symbols and background with hollow symbols

and sequential transports that need to be performed. Recall that the number of sequential transports is the only non-parallel part of our implementation.

Three sets of experiments have been taken into account to test the discriminatory power, efficiency and practical use of the proposed representation.

5.1 Experiments testing the *Hom-Tree* topological discrimination power

A simple object representing a foreground ring normal to the Z axis is shown in Fig. 5. There is one 0-cell (representative of the foreground connected component) and one 1-cell (representative of the foreground tunnel). Correspondingly, the background ambience that surrounds the ring contains one 0-cell (represented by the most upper right 0-cell), one 1-cell (the foreground ring is seen like a tunnel for the background) and a 2-cell, representative of the background cavity (which is indeed the foreground ring). The fact that each critical cell has its corresponding dual (except the background component of the canvas) implies that the Euler-Poincaré number of the whole 3D image is equal to 1. This fact shows the ability of the *Hom-Tree* representation to topologically classify images.

An automatic homotopy deformation tool (a thinning-thickening software based on the notion of a simple 3D point [16], and developed for this purpose as detailed in [8]), has been used to support experimentation. Synthetic images have been randomly deformed using such a deformation tool, validating the isotopy-invariance property of the *Hom-Tree*. This

software corroborates that homotopically (in fact, isotopically) equivalent shapes return the same *Hom-Tree*. Examples of these experiments are included in Appendix B.

The *Hom-Tree* representation is also able to distinguish non-isotopic patterns having the same homology (a sphere with two inner handles against a sphere with two outer handles, a ring inside a sphere against a sphere surrounded by a ring, etc.). It is worth mentioning that some dissimilar nesting of digital objects cannot be distinguished using the *Hom-Tree* representation: this is the case of a torus versus a sphere with an inner tunnel and an external one. Both result in the same *Hom-Tree* representation and edge weights. Further considerations in this matter are discussed in Section 6.

5.2 Experiments testing parallelism and efficiency of the *Hom-Tree* construction

In this Section experiments were performed using modifications of 3D Menger Sponges of different dimensions. The initial structures used to generate these new fractals are constructed filling a few planes (OX, OY and/or OZ) within the Menger Sponges of recursion 2 and 3 for different values of x , y and z . A variation of the Menger sponge of recursion level 2 in which the OX plane for $x=1$, the OY plane for $y=3$ and the OZ plane for $z=1$ are filled; and a variation of the Menger sponge of recursion level 3, in which the OX plane for $x=1$, the OY plane for $y=3$ and the OZ plane for $z=1$ are filled. In these cases, some transports to cancel (1, 2)-pairs of false critical cells are necessary. The detailed results can be seen in Appendix B. In order to check the correctness of our implementation, we validate the Euler-Poincaré characteristic of different Menger sponges. Note that the Euler-Poincaré characteristic can be directly extracted for the number of cells in every column of each table, due to the fact that the HSF building has previously divided the different objects of the image into separated trees. For example, Euler-Poincaré characteristic for the first Menger sponge 2X1Y3Z1 results to be: $1 - 28 + 10 = -17$. Evidently, this number remains constant along the different parallel and sequential stages until the final HSF is completed. Then, Betti numbers are obtained in this column of the final HSF. In all cases, the first parallel transport iteration of (0, 1)-cell pairs yielded the correct result. However, the results for the (1, 2)-pairs were very divergent. On the one hand, for some sponges the final HSFs are obtained through our parallel transportation method; on the other hand, for some others, the parallel transports produce moderate results (around fifty per cent of sequential (1, 2)-pair transports were required). Table 1 presents a summary of the results for three modified sponges.

There is no doubt that synthetic images are homologically complex and the parallel processing of an HSF can be more expensive. On the contrary, parallel computing is very efficient for random images, and even more so for real medical images. For the ten small random images (up to $15 \times 15 \times 15$) tested, the results were not very meaningful: The percentage of parallel transports for the (1, 2)-pairs were always a 100% and only one image presented a percentage of 97% for the (0, 1)-pairs.

For bigger images, some sequential transport iterations were needed. We refer the reader to the Appendix B (Tables 2 and 3) to consult the mean number of critical cells for the first four parallel and last sequential transport iterations of (0, 1)-cell pairs, and for the following parallel and final sequential transport of (1, 2)-cell pairs, given a set of five $20 \times 20 \times 20$ and $30 \times 30 \times 30$ B/W random images of 50% density, respectively.

The last three columns reveal patently the efficiency of our parallel processing: the first phase of parallel (0, 1)-transports reached more that three quarters of the total necessary transports. After the fourth parallel stage, this percentage reaches more than 90%, which

Table 1 Number of critical cracks during the first parallel transport iteration of (0, 1)-cell pairs and at the end of the last sequential phase for two variations of the Menger sponge

	Initial critical cells (MrSF)	After 1st transports of 0/1 cells	After 2nd transports of 0/1 cells	After sequential transports of 0/1 cells	After parallel transports (1/2 cells)	After sequential transports of 1/2 cells (final HSF)	Percentage of parallel transports (0/1 cells)	Percentage of parallel transports (1/2 cells)
FG critical 0-cells	1	1	1	1	1	1	-	-
FG critical 1-cells	28	28	28	28	23	19	-	55.55
FG critical 2-cells	10	10	10	10	5	1	-	55.55
FG critical 0-cells	1	1	1	1	1	1	-	-
FG critical 1-cells	852	852	852	852	698	584	-	57.46
FG critical 2-cells	262	262	262	262	108	0	-	57.46
	Initial critical cells (MrSF)	After 1st transports of 0/1 cells	After parallel transports (1/2 cells)	After sequential transports of (1/2 cells) (final HSF)	Percentage of parallel transports (0/1 cells)	Percentage of parallel transports (1/2 cells)		
FG critical 0-cells	1	1	1	1	-	-	-	-
FG critical 1-cells	1156	1156	915	915	-	100.00	-	-
FG critical 2-cells	241	241	0	0	-	100.00	-	-

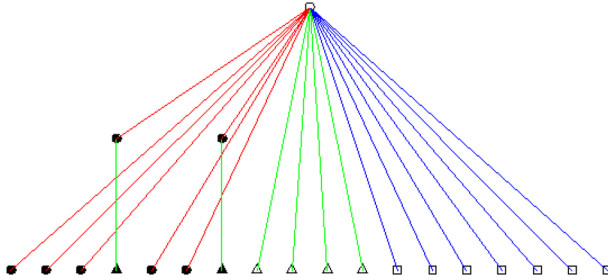


Fig. 10 Left: Unweighted *Hom-Tree* for a $30 \times 30 \times 30$ fragment of a medical image of PET used as an example

supposes a sequential number of steps that is less than 0.5% of the total number of voxels for any random image. The results about (1, 2) pairs are even better: almost all couples can be canceled in parallel.

5.3 Experiments using real images

The third set of tests was performed for three real binary 3D medical images. Clinical Micro Computer Tomography Images of trabecular bones (obtained by the ETH, Zurich) were selected due to their typically complex topology, consisting of many cavities and tunnels (see Table 4 in the Appendix B). The results are similar to those of random images, that is, around a 90% of (0, 1)-pair transports and around 97% of (1, 2)-pair transports can be done in a fully parallel manner just in four iterations for the first pairs and in one iteration for the second ones. Because natural images are usually simpler than random ones, fewer transports are necessary for them. In fact, the sequential number of steps for this set of medical images is around ten times inferior (less than 0.05% of the total amount of voxels).

The presence of critical 1 and 2-cells is very common in many 3D medical images, which present a good number of tunnels and cavities (see Fig. 10). Due to the high resolution of the employed data, which correspond to a PET (Positron Emission Tomography) image of a person’s head, downsampling over the pixels was performed with a factor of 6, followed by a segmentation from applying Otsu’s method in order to obtain the final binary image. Despite decreasing the resolution of the image, interesting results were obtained. Thus, bounding relations among critical cells may serve in the future to analyze the complex nature of these images.

6 Conclusions

In this paper, the design of a new parallel algorithm for computing the *Hom-Tree* representation model of a 3D binary digital (6, 26)-image is presented. This *Hom-Tree* representation codifies the topological duality relationships among homology characteristics via a cellular complex analogous to the image. A complete implementation of the proposed algorithm has been performed in the MATLAB/OCTAVE language. The results show that its time complexity order is almost logarithmic (only remaining a linear term less than 0.5% of the total amount

of voxels even for random images). The usability of the *Hom-Tree* model is shown through examples in which its ability to homologically, homotopically and isotopically differentiate between similar objects is demonstrated.

The potential of this new representation opens new research lines that will be addressed in future work. We plan to advance in the construction of new graph-based representation models improving the topological discriminatory power of the *Hom-Tree* model in a 3D binary image context (i.e. classification of complex topological structures like knots) and to properly extend the *Hom-Tree* model to color and n -dimensional digital images. In this last respect, the extension of the HSF algorithm and the *Hom-Tree* representation to 4D data will allow analyzing how topological structures evolve over time and how they change or conserve their previous interactions. Likewise to the Definition 1 and for the case of higher dimensions, the *Hom-Tree* weight of an edge (R, S) , (being R and S two nested regions and, consequently, one is necessarily a black $(2n - 1)$ -CC and the other one is a white $(2^n - 1)$ -CC), is a vector composed of the numbers of k -dimensional homological holes “shared” by R and S , being $1 \leq k \leq (n - 2)$.

Functional magnetic resonance images (fMRI) are a well-known example of how interactions among topological structures are beginning to be studied. For example, suppose that a 4^{th} dimensional canvas for a given window time t be C_t^4 , where every 3D image will be a subset for a given temporal sample, that is, $C_t^3 \subset C_t^4$. A healthy patient undertaking a stimulation test exhibits a brain activity pattern that generates a temporal sequence of 3D images. That is, the 4^{th} dimension could allow to track the spatial distribution changes between consecutive times, i.e, the time-series given by: $C_{t_0}^3 - C_{t_1}^3 - C_{t_2}^3 - \dots - C_{t_n}^3$. The resulting set of all the computed topological information, comprised of the different *Hom-Trees* for every 3D temporal frame, opens a new field of machine learning approaches focused on processing the compressed topological *Hom-Trees* instead of the complete and huge 4-dimensional images [39]. This condensed topological information may help find disease markers that allow for the diagnosis and detection of anomalies within these and other application contexts.

A *Hom-Tree* hierarchical system following the boundary scale-space model introduced in [40] is also to be developed. An implementation in C++ is feasible and will be performed in the near future, by slightly modifying our previous HSF implementation published in [41]. The complicated programming of HSF trees in a parallel fashion points in the direction that a C++ version using the paradigm of an inherent parallel Cellular Automata, might be the most practical solution to release a fast and efficient *Hom-Tree* implementation. Finally, *Hom-Tree* representation may be incorporated into the field of 3D image machine learning as an efficient image preprocessing step, providing topological information to the process.

Appendix

A parallel generation of Hom-Tree. HSF construction via crack transports

For each critical MrSF 0-crack, we calculate in parallel the movements along $(1, 2)$ and $(0, 1)$ trees. Firstly, there are six possible paths leaving from each critical 0-cell and moving through the $(1, 2)$ -tree. One of them is selected for every critical MrSF 0-crack, and the $(1, 2)$ -trees (tracking frontier digital surfaces), fall onto the corresponding 2-cells (black arches in Fig. 11). From these 2-cells, it is resolved what critical 1-crack has a primal vector to them.

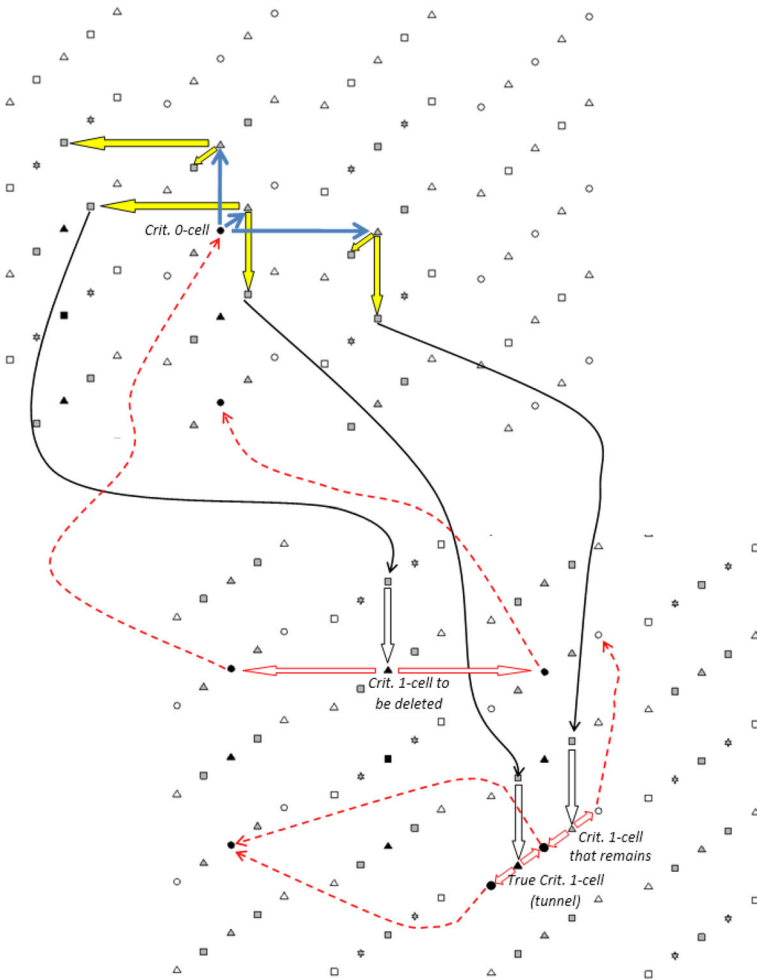


Fig. 11 The six possible paths from a critical MrSF 0-cell moving through the $(1, 2)$ -tree to determine if this 0-cell can be paired with a 1-cell in parallel. The higher black 0-cell is a critical MrSF 0-cell, having three incident 1-cells (going towards positive directions, drawn with blue darts). Every one of these 1-cells has two incident 2-cells (going towards negative directions, drawn with yellow arrows). Then, black arches depict movements along the $(1, 2)$ -tree. These movements begin tracking a digital surface frontier, and finally fall onto 2-cells. From these 2-cells, the critical 1-cells that have a primal vector to them are shown (see white arrows). Then, the back moves are computed in order to determine if the couple 0-cell, 1-cell can be paired. Every 1-cell has two incident 0-cells (red arrows). Thus, movements of these 0-cells through the $(0, 1)$ -tree yield three different cases: 1) the 1-cell can be deleted (a crack transport is done) with the initial critical 0-cell, because only one of its two $(0, 1)$ -paths fell into the initial 0-cell; 2) the 1-cell is the representative of a tunnel (it is a true critical 1-cell), since these two $(0, 1)$ paths fell into the same 0-cell; 3) Nothing can be determined, because the two $(0, 1)$ paths fell into two different 0-cells (so the 1-cell remains the same)

Then, every 1-cell has two incident 0-cells (red arrows in Fig. 11), and the two back movements for these 0-cells are computed in parallel in order to determine if the couple of (0, 1)-cells can be eliminated, and so the transport is carried out. The movements of these 0-cells through the (0, 1)-tree yield three different cases (processed in an independent and parallel manner for each couple of (0, 1)-cells). These cases are: a) the 1-cell can be deleted (a crack transport is carried out) with the initial critical 0-cell, because only one of its two (0, 1)-paths fell into the initial 0-cell; b) the 1-cell is the representative of a tunnel (it is a true critical 1-cell), since these two (0, 1) paths fell into the same 0-cell; c) nothing can be determined, because the two (0, 1)-paths fell into two different 0-cells (in this case, the 1-cell remains the same).

Note that moving along trees guarantees that the uniqueness in the selection of the different couples of (0, 1)-cells is ensured, and therefore there is no need for synchronization primitives. In this sense, perfect parallelism is achieved in these six paths, thus, their time order can be considered as $O(1)$. Summing up, for each critical MrSF 0-crack, six set of paths must be computed (all the paths of each set in parallel). Likewise, for the pairing of couples of critical MrSF 1-crack, a similar parallel procedure can be performed, preserving the time order as $O(1)$. The number of incident cells is inferior in this ulterior pairing. More concretely, each critical MrSF 1-cell has only two possible paths that go through (2, 3)-trees. These trees would fall into 3-cells that have associated (through primal vectors) with a possible critical 2-cell. Each of these 2-cells has up to four 1-cells in their borders; thus, four back movements along (1, 2)-trees must be then calculated. A comparison of the 1-cells where the back movements fall will result in these cases: 1) the 2-cell may be the representative of a cavity (a true critical HSF 2-cell), since at least two of them fell into the initial critical MrSF 1-cell; 2) the 2-cell can be paired with the initial critical MrSF 1-cell, because only one of the back travel fell into this last; 3) nothing can be determined, because the two previous cases did not occur.

B Further experimental results

Figure 12 shows examples of deformed objects generated with the automatic homotopy deformation (thinning-thickening) software, based on the notion of a simple 3D point. Up to 500 26-simple points have been randomly added/removed 100 times for each of these cases: Three concentric spheres, one torus with two spheres inside, and one sphere with two spheres inside. These examples were used to corroborate that homotopically (in fact, isotopically) equivalent nesting return the same *Hom-Tree*.

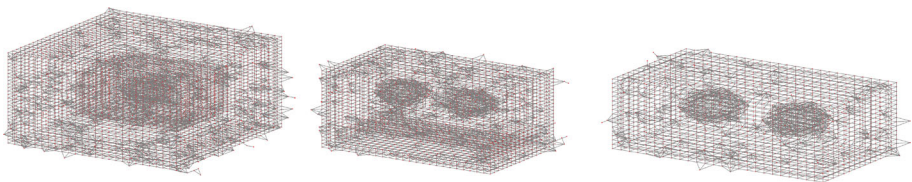


Fig. 12 Three examples of deformed objects: Three concentric spheres, one torus with two spheres inside, and one sphere with two spheres inside

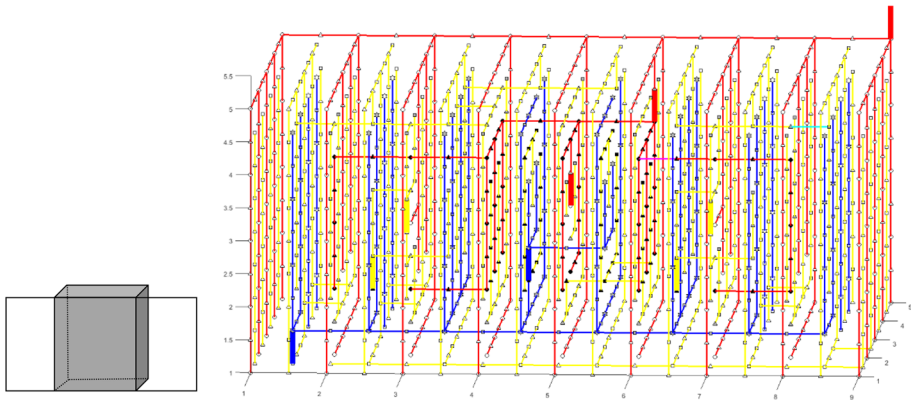


Fig. 13 HSF of a sphere with two external handles. The color code is the same as in the paper. All trees are displayed along with their corresponding critical cells. In the upper right part of the figure, a red link represents the background CC of the image

In Fig. 13 an HSF of a foreground sphere with two external handles is completely drawn to distinguish that the borders of its two critical 1-cells (marked with thickest yellow lines at $Z = 2, Y = 2, X = 2.5$ and $X = 6.5$) fall into the foreground component. Moreover, there are two additional tunnels for the external background component (ahead of the two previous ones at $Y = 1.5$).

Tables 2 and 3 show the mean number of critical cells for the first four parallel and last sequential transport iterations of $(0, 1)$ -cell pairs, and for the following parallel and final sequential transport of $(1, 2)$ -cell pairs for a set of five $20 \times 20 \times 20$ and $30 \times 30 \times 30$ B/W random images, resp. These images were generated using a 50/50 probability for colors, and surrounded by six thin faces of background voxels according to our working premises. Actually, a fifth iteration would transport another one or two $(0, 1)$ -pairs more for two of the bigger images tested.

Table 4 presents a summary of the results for the images of the trabecular bone.

The *Hom-Tree* is able to distinguish non-isotopic patterns having the same homology. An interesting example occurs when a ring inside a sphere is contrasted against a sphere surrounded by a ring (see Figs. 14 and 15)(Fig. 16).

Table 2 Evolution of the mean number of critical cells for a set of five BW $20 \times 20 \times 20$ random images with a 50/50 probability for colors

	Initial critical cells (MrSF)	After 1st transports of 0/1 cells	After 2nd transports of 0/1 cells	After 3rd transports of 0/1 cells	After 4th transports of 0/1 cells	After sequential transports of 0/1 cells	After parallel transports of 1/2 cells (final HSF)	After sequential transports of 1/2 cells in the 1st transport (0/1 cells)	Percentage of parallel transports (0/1 cells)	Percentage of parallel transports (1/2 cells)
FG critical 0-cells	436.20	182.60	135.80	117.00	114.40	80.40	80.40	80.40	78.81	90.44
FG critical 1-cells	706.20	452.60	405.80	387.00	384.40	350.40	330.80	330.80	78.81	90.44
FG critical 2-cells	19.60	19.60	19.60	19.60	19.60	19.60	0.00	0.00	0.00	100.00

The images were later surrounded by six thin faces of background voxels

The first column contains the initial number of critical cells

Columns 3 to 7 show the evolution of the foreground critical cells for the first four parallel and last sequential transport iterations of (0, 1)-cell pairs

Columns 8 and 9 show the evolution for the following parallel and final sequential transport of (1, 2)-cell pairs

The last three columns indicate the mean percentage of parallel transports achieved: in the first iteration for (0, 1)-pairs, after the fourth iteration for (0, 1)-pairs, and for the (1, 2)-pair transport, resp

Table 3 Evolution of the mean number of critical cells for a set of five B/W $30 \times 30 \times 30$ random images with a 50/50 probability for colors

	Initial critical cells (MrSF)	After 1st trans-ports of 0/1 cells	After 2nd trans-ports of 0/1 cells	After 3rd trans-ports of 0/1 cells	After 4th trans-ports of 0/1 cells	After sequential trans-ports of 0/1 cells	After parallel trans-ports of 1/2 cells (final HSF)	After sequential trans-ports of 1/2 cells (final HSF)	Percentage of parallel transports in the 1st transport (0/1 cells)	Percentage of parallel transports (0/1 cells)	Percentage of parallel transports (1/2 cells)
FG critical 0-cells	1510.60	620.20	444.20	370.40	358.40	241.60	241.60	241.60	77.28	90.80	
FG critical 1-cells	2746.40	1856.00	1680.00	1606.20	1594.20	1477.40	1401.00	1400.20	77.28	90.80	98.96
FG critical 2-cells	77.20	77.20	77.20	77.20	77.20	77.20	0.80	0.00			98.96

The images were later surrounded by six thin faces of background voxels

The first column contains the initial number of critical cells

Columns 3 to 7 show the evolution of the critical foreground cells for the first four parallel and last sequential transport iterations of (0, 1)-cell pairs

Columns 8 and 9 show the evolution for the following parallel and final sequential transport of (1, 2)-cell pairs

The last three columns indicate the mean percentage of parallel transports achieved: in the first iteration for (0, 1)-pairs, after the fourth iteration for (0, 1)-pairs, and for the (1, 2)-pair transport, resp

Table 4 Evolution of the number of critical cracks for three trabecular bone images of sizes 43x43x9, 64x64x13, and 43x43x9 (rows 1-3, 4-6 and 7-9 respectively)

	Initial critical cells (MrSF)	After 1st transports of 0/1 cells	After 2nd transports of 0/1 cells	After 3rd transports of 0/1 cells	After 4th transports of 0/1 cells	After sequential transports of 0/1 cells	After parallel transports of 1/2 cells	After sequential transports of 1/2 cells (final HSF)	Percentage of parallel transports in the 1st transport (0/1 cells)	Percentage of parallel transports (0/1 cells)	Percentage of parallel transports (1/2 cells)
FG critical	71	19	14	12	11	6	6	6	86.67	92.31	
0-cells											
FG critical	148	96	91	89	88	83	21	19	86.67	92.31	96.88
1-cells											
FG critical	106	106	106	106	106	106	44	42			96.88
2-cells											
FG critical	187	77	60	46	40	16	16	16	74.83	85.96	
0-cells											
FG critical	242	132	115	101	95	71	14	14	74.83	85.96	100.00
1-cells											
FG critical	60	60	60	60	60	60	3	3			100.00
2-cells											

Table 4 continued

	Initial critical cells (MrSF)	After 1st transports of 0/1 cells	After 2nd transports of 0/1 cells	After 3rd transports of 0/1 cells	After 4th transports of 0/1 cells	After sequential transports of 0/1 cells	After parallel transports of 1/2 cells	After sequential transports of 1/2 cells (final HSF)	Percentage of parallel transports in the 1st transport (0/1 cells)	Percentage of parallel transports (0/1 cells)	Percentage of parallel transports (1/2 cells)
FG critical 0-cells	102	37	28	19	16	8	8	8	75.58	91.49	
FG critical 1-cells	190	125	116	107	104	96	28	26	75.58	91.49	97.14
FG critical 2-cells	101	101	101	101	101	101	33	31			97.14

The table includes the first four parallel and last sequential transport iterations of (0, 1)-cell pairs, and the following parallel and final sequential transport of (1, 2)-cell pairs. The last three columns indicate the mean percentage of parallel transports achieved: in the first iteration for (0, 1)-pairs, after the fourth iteration for (0, 1)-pairs, and for the (1, 2)- pair transport, resp

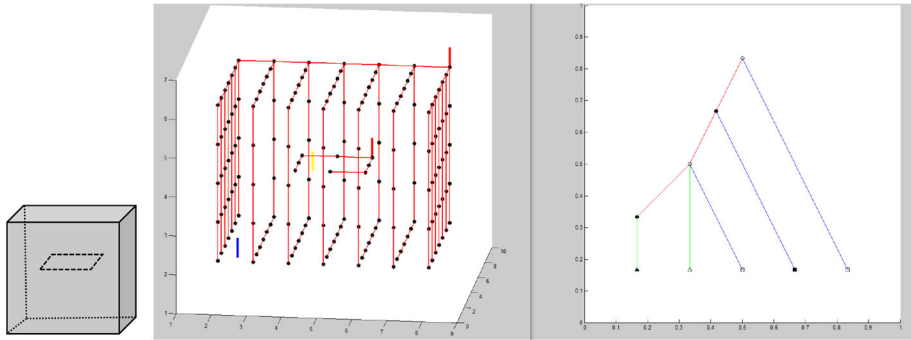


Fig. 14 Left: foreground (0, 1)-Tree for a ring inside a sphere. The same color convention is followed. Only foreground 0, 1 and 2-cells are shown. Critical cells are marked with a gross vertical segment. Right: *Hom-Tree* for this object

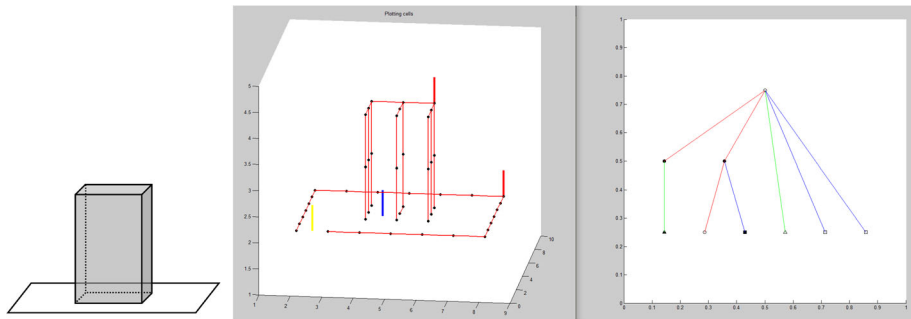


Fig. 15 Left: foreground (0, 1)-Tree for a sphere inside a ring. The same color convention is followed. Only foreground 0, 1 and 2-cells are shown. Critical cells are marked with a gross vertical segment. Right: *Hom-Tree* for this object

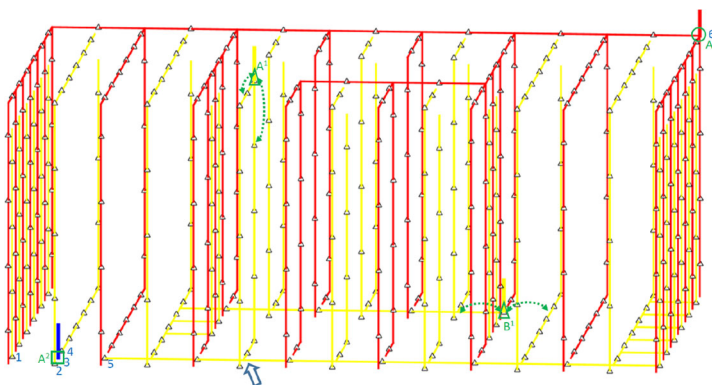


Fig. 16 HSF of a torus. Only 1-cells (empty triangles) are plotted for the sake of clarity. Critical 0, 1, 2-cells are marked with green circles, triangles and squares resp. Only those relevant sub-graphs to detect boundaries are labeled (in fact, some sub-graphs are very short due to pruning to prevent cycles when building the HSF). The blue arrow points to an inactive edge (due to pruning) that would determine a cycle in sub-graph 5. Note that sub-graph 5 holds both critical 1-cells (marked with green triangles). Green dashed arrows indicate the boundary relations that compound a cycle inside sub-graph 5.

Acknowledgements This work has been partially supported by the research projects Par-Hot (PID2019-110455GB-I00) and CIUCAP-HSF (US-1381077) funded by MCIN/AEI/10.13039/501100011033 and Feder Funds.

Funding Funding for open access publishing: Universidad de Sevilla/CBUA.

Data availability statement(DAS) Data sharing not applicable.

Declarations

Conflict of interest/Competing interests Not applicable

Open Access This article is licensed under a Creative Commons Attribution 4.0 International License, which permits use, sharing, adaptation, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons licence, and indicate if changes were made. The images or other third party material in this article are included in the article's Creative Commons licence, unless indicated otherwise in a credit line to the material. If material is not included in the article's Creative Commons licence and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright holder. To view a copy of this licence, visit <http://creativecommons.org/licenses/by/4.0/>.

References

1. Hensel, F., Moor, M., Rieck, B.: A survey of topological machine learning methods. *Front. Artif. Intell.* **4**, 52 (2021)
2. Pun, C.S., Lee, S.X., Xia, K.: Persistent-homology-based machine learning: a survey and a comparative study. *Artif. Intell. Rev* (2022)
3. Sanfeliu, A., Alquézar, R., Andrade, J., Climent, J., Serratos, F., Vergés, J.: Graph-based representations and techniques for image processing and image analysis. *Pattern Recognit.* **35**(3), 639–650 (2002)
4. Klette, R.: Cell complexes through time. In: *Vision Geometry IX*, SPIE, vol. 4117, pp. 134–145. (2000)
5. Pavlidis, T.: *Structural pattern recognition*. Springer-Verlag, New York (1977)
6. Rosenfeld, A.: Adjacency in digital pictures. *Inf. Control* **26**(1), 24–33 (1974)
7. Serra, J.: *Image analysis and mathematical morphology*. Press, Acad (1982)
8. Real, P., Molina-Abril, H., Díaz-del-Río, F., Blanco-Trejo, S.: Homological region adjacency tree for a 3D binary digital image via HSF model. In: *Int. Conf. CAIP19*, pp. 375–287. Springer, Cham (2019)
9. Molina-Abril, H., Real, P.: Homological spanning forest framework for 2D image analysis. *Ann. Math. Art. Int.* **64**(4), 385–409 (2012)
10. Molina-Abril, H., Real, P., Nakamura, A., Klette, R.: Connectivity calculus of fractal polyhedrons. *Pattern Recognit.* **48**(4), 1150–1160 (2015)
11. Latecki, L.J.: 3d well-composed pictures. *Graphical Models and Image Processing.* **59**(3), 164–172 (1997)
12. Cerman, M., Janusch, I., González-Díaz, R., Kropatsch, W.G.: Topology-based image segmentation using LBP pyramids. *Mach. Vis. Appl.* **27**(8), 1161–1174 (2016)
13. Delgado-Friedrichs, O., Robins, V., Sheppard, A.: Skeletonization and partitioning of digital images using discrete morse theory. *IEEE Trans. Pattern Anal Mach Intell.* **37**(3), 654–666 (2015)
14. Damiand, G., Lienhardt, P.: *Combinatorial Maps: Efficient Data Structures for Computer Graphics and Image Processing*. CRC Press, Boca Ratón (2014)
15. Abrams, L., Fishkind, D.E.: The genus of a digital image boundary is determined by its foreground, background, and reeb graphs. *Discret. Comput. Geom.* **37**(4), 629–640 (2007)
16. Bertrand, G.: Simple points, topological numbers and geodesic neighborhoods in cubic grids. *Pat. Rec. Let.* **35**, 1003–1011 (1994)
17. Klette, R.: *Skeletons in digital image processing*. Glen Innes: Comput. Sci. Dept. University of Auckland **21** (2002)
18. Diaz-del-Rio, F., Real, P., Onchis, D.M.: A parallel homological spanning forest framework for 2D topological image analysis. *Pat. Rec. Let.* 49–58 (2016)
19. Diaz-del-Rio, F., Sanchez-Cuevas, P., Molina-Abril, H., Real, P.: Parallel connected-component-labeling based on homotopy trees. *Pat. Rec. Let.* **131**, 71–78 (2020)

20. Romero, A., Rubio, J., Sergeraert, F.: Effective homology of filtered digital images. *Pat. Rec. Let.* **83**, 23–31 (2016)
21. González-Díaz, R., Real, P.: On the cohomology of 3D digital images. *Discret. Appl. Math.* **147**(2–3), 245–263 (2005)
22. Pilarczyk, P., Real, P.: Computation of cubical homology, (co)homology and (co)homological operations via chain contractions. *Adv. Comput. Math.* **41**(1), 253–275 (2015)
23. Delgado-Friedrichs, O., Robins, V., Sheppard, A.: Morse theory and persistent homology for topological analysis of 3D images of complex materials. *IEEE Int. Conf. Image Process. (ICIP)* 4872–4876 (2014)
24. Chung, Y.M., Day, S.: Topological fidelity and image thresholding: A persistent homology approach. *J. Math. Imaging Vision* **60**(7), 1167–1179 (2018)
25. Kong, T.Y., Rosenfeld, A.: Digital topology: Introduction and survey. *Comput. Vision Graphics Image Process.* **48**(3), 357–393 (1989)
26. Kong, T.Y., Roscoe, A.W., Rosenfeld, A.: Concepts of digital topology. *Topol. Appl.* 219–262 (1992)
27. Khachan, M., Chenin, H.P.: Deddi: Polyhedral representation and adjacency graph in n-dimensional digital images. *Comput. Vision Image Underst.* **79**(3), 428–441 (2000)
28. Kovalevsky, V.: *Modern Algorithms for Image Processing: Computer Imagery by Example Using C*. Apress, New York (2018)
29. Gray, S.B.: Local properties of binary images in two and three dimensions. *IEEE Trans. Comput.* **20**(5), 551–561 (1970)
30. Lee, C.N., Rosenfeld, A.: Computing the euler number of a 3d image. *Proc. IEEE First Int. Conf. Comput. Vision* 567–571 (1987)
31. Munkres, J.R.: *Elements of Algebraic Topology*. Addison Wesley, California (1984)
32. Kovalevsky, V.: Algorithms in digital geometry based on cellular topology. 10th IWCIA, Springer, Berlin Heidelberg **3322**, 366–393 (2004)
33. Lundell, A.T., Weingram, S.: *The Topology of CW-complexes*. Van Nostrand Reinhold, Springer, New York (1969)
34. Real, P., Díaz-del-Río, F., Onchis, D.: Toward parallel computation of dense homotopy skeletons for nD digital objects. *Combinatorial Image Analysis. IWCIA 2017. LNCS.* **10256** (2017)
35. Forman, R.: Morse theory for cell complexes. *Adv. Math.* **134**, 90–145 (1998)
36. Komura, Y.: GPU based cluster-labeling algorithm without the use of conventional iteration: Application to the Swendsen-Wang multi-cluster spin flip algorithm. *Comput. Phys. Commun.* **194**, 54–58 (2015)
37. Díaz-del-Río, F., Blanco-Trejo, S., Real, P., Molina-Abril, H., Onchis, D.: HSF and Hom-Tree of 3D images. <https://www.mathworks.com/matlabcentral/fileexchange/80503-hsf-and-hom-tree-of-3d-images>. Accessed March 25, 2022 (2022)
38. Gonzalez-Lorenzo, A., Bac, A., Mari, J.-L.: A heuristic for short homology basis of digital objects. In: Baudrier, É., Naegel, B., Krähenbühl, A., Tajine, M. (eds.) *Discrete Geometry and Mathematical Morphology*, pp. 60–70. Springer, Cham (2022)
39. Bukkuri, A., Andor, N., Darcy, I.K.: Applications of topological data analysis in oncology. *Front. Arti. Intell.* **4**, 38 (2021)
40. Molina-Abril, H., Real, P., Díaz-del-Río, F.: Generating (co) homological information using boundary scale. *Pat. Rec. Let.* **133**, 240–246 (2000)
41. Díaz-del-Río, F., Sánchez-Cuevas, P., Molina-Abril, H., Real, P.: Parallel connected-component-labeling based on homotopy trees. *Pat. Rec. Let.* **131**, 71–78 (2000)

Publisher's Note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.

Authors and Affiliations

Fernando Díaz-del-Río¹ · Helena Molina-Abril² · Pedro Real¹  · Darian Onchis³ · Sergio Blanco-Trejo⁴

Fernando Díaz-del-Río
fdiaz@us.es

Helena Molina-Abril
habril@us.es

Darian Onchis
darian.onchis@e-uvt.ro

Sergio Blanco-Trejo
sblanco1@us.es

- ¹ Institute of Informatics Engineering (I3US), University de Sevilla, Avda. Reina Mercedes s/n, Sevilla 14012, Spain
- ² Institute of Mathematics (IMUS), University de Sevilla, Avda. Reina Mercedes s/n, Sevilla 14012, Spain
- ³ Faculty of Mathematics and Informatics, West University of Timisoara, Street Vasile Pârvan no. 4, Timisoara 300223, Romania
- ⁴ Escuela Técnica Superior de Ingenieros, Universidad de Sevilla, Avda. de los Descubrimientos s/n, Sevilla 41092, Spain