Journal of Mathematical Imaging and Vision (2023) 65:915–930
https://doi.org/10.1007/s10851-023-01153-y

Surface-Based Computation of the Euler Characteristic in the BCC Grid

Lidija Čomić[1] · **Paola Magillo**[2]

Received: 24 September 2022 / Accepted: 14 June 2023 / Published online: 17 July 2023

Abstract
As opposed to the 3D cubic grid, the body-centered cubic (BCC) grid has some favorable topological properties: each set of voxels in the grid is a 3-manifold, with 2-manifold boundary. Thus, the Euler characteristic of an object O in this grid can be computed as half of the Euler characteristic of its boundary ∂O. We propose three new algorithms to compute the Euler characteristic in the BCC grid with this surface-based approach: one based on (critical point) Morse theory and two based on the discrete Gauss–Bonnet theorem. We provide a comparison between the three new algorithms and the classic approach based on counting the number of cells, either of the 3D object or of its 2D boundary surface.

Keywords Digital topology · Euler characteristic · Body Centered Cubic (BCC) grid · Morse theory · Discrete Gauss-Bonnet theorem

1 Introduction

The Euler characteristic is a basic yet important, locally computable topological descriptor of both continuous and discrete shapes. It is equal to the alternating sum of (independent) i-dimensional holes. For shapes in 3D, it is equal to the number of connected components minus the number of tunnels plus the number of cavities. It can be computed as the alternating sum of the number of i-cells in a cell decomposition of the shape.

The commonly used 3D cubic grid has some unfavorable geometric and topological properties, mainly due to the existence of three different types of adjacencies between the cubes. This causes many image processing algorithms to become more involved, needing more computation [22]. The body-centered cubic (BCC) grid is one feasible alternative to the traditional 3D cubic one, with many topological advantages. It has been effectively used in computer graphics [3, 10, 13], discrete geometry [9], tomographic reconstruction from projections [4], physically-based simulations [25], voxelization [15], distances by neighborhood sequences [36] and in various other fields including discretization, ray tracing and ray casting, volume rendering and repairing [7, 10, 13, 16, 17, 32]. A software system for processing and viewing 3D data on this grid has also been developed [29].

The BCC grid is a tessellation of the space $\mathbb{R}^3$ through (space-filling) truncated octahedra. Two voxels in the BCC grid are either disjoint, or they share a whole face. This ensures that the boundary ∂O of an object (a finite set of voxels) O in the BCC grid is always a 2-manifold (and O is a 3-manifold). Therefore, the Euler characteristic of O can be computed as half the Euler characteristic of its boundary ∂O, which is likely to have significantly fewer cells than O in many practical cases.

Here, we address the computation of the Euler characteristic in the BCC grid with this surface-based approach. Apart from the known algorithm based on counting the boundary cells, we propose to use two frameworks for computing the Euler characteristic of a polyhedron (the boundary ∂O of an object O) in this grid.

The first framework is the piecewise linear [1] critical point Morse theory [30, 33], which provides relationships between the topology of a manifold M and the critical points of a scalar field f defined on M. The second is the discrete version of the Gauss–Bonnet theorem, which states that the

This work has been partially supported by the Ministry of Science, Technological Development and Innovation of the Republic of Serbia through the Project No. 451-03-47/2023-01/200156.

✉ Paola Magillo
magillo@dibris.unige.it

Lidija Čomić
comic@uns.ac.rs

[1] Faculty of Technical Sciences, University of Novi Sad, Novi Sad, Serbia

[2] Department of Computer Science, Bioengineering, Robotics, and Systems Engineering, University of Genova, Genova, Italy

sum of angular vertex deficits in a polyhedron is equal to 2π times its Euler characteristic. We propose two algorithms based on this framework.

We provide a detailed comparison of the proposed algorithms, and a comparison of them with the algorithm implementing the classic approach based on counting the number of cells, in a volume-based or surface-based version (i.e., on the 3D object or on its 2D boundary).

In particular, one of the two new algorithms based on the Gauss–Bonnet theorem is the fastest surface-based method, with execution times equal to about 40% of the naive 2D cell counting. The surface-based approach to the Euler characteristic computation is convenient over the volume-based one when the number of boundary faces is less than 25% (if we include the cost of boundary extraction) or less than 66% (if we consider only the Euler characteristic computation) than the number of voxels composing the object.

2 Background Notions

We give some basic notions on 3D digital topology [20, 24], the BCC grid [19], the Euler characteristic [31], Morse theory [30, 33] and the discrete Gauss–Bonnet theorem [23, 35].

2.1 Basic Notions

A k-cell in $\mathbb{R}^n$ (with $0 \leq k \leq n$) is a homeomorphic image of the closed unit k-ball $\{p \in \mathbb{R}^k : ||p|| \leq 1\}$. A cell k-complex Γ is a collection of i-cells, with $\max\{i\} = k$, that fit nicely together: the boundary of each cell and the intersection of any two cells (if non-empty) is composed of cells of lower dimension. We are interested in the cases $n = 3$ and $k = 2, 3$ that are used to represent solid objects and surfaces in $\mathbb{R}^3$.

A subset $I \subseteq \mathbb{R}^n$ is a k-manifold (with boundary) if each of its points has a neighborhood homeomorphic to $\mathbb{R}^k$ (or to $\mathbb{R}^k_+ = \{(x_1, \ldots, x_k) \in \mathbb{R}^k : x_k \geq 0\}$). In $\mathbb{R}^3$, we are interested in 3-manifolds with boundary, representing a solid object, and 2-manifolds, representing surfaces. The boundary surface of a 3-manifold object is a (closed) 2-manifold without boundary.

The set of points in a cell complex Γ in $\mathbb{R}^3$ is denoted as $|\Gamma|$. If $|\Gamma|$ is a manifold, we say that Γ is a manifold cell complex.

2.2 The Cubic Grid

The 3D cubic grid is the regular tessellation of $\mathbb{R}^3$ into closed unit cubes centered at points in $\mathbb{Z}^3$, with faces parallel to the coordinate planes [20]. Three types of adjacency relation are defined between the cubes in the grid, depending on their intersection. Two cubes are face-, edge-, or vertex-adjacent if they share at least one face, edge or vertex, respectively. They are strictly edge- or vertex-adjacent if they are edge- or vertex-adjacent but are not face- or edge-adjacent, respectively. Connectivity relation is the transitive closure of adjacency.

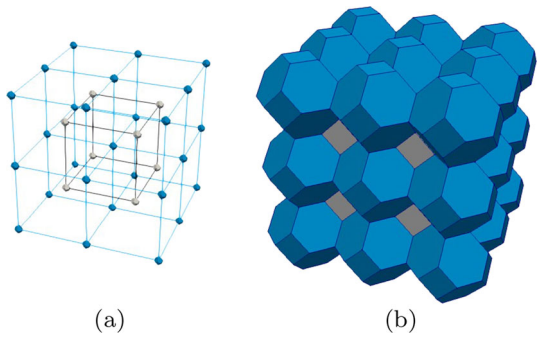


Fig. 1 **a** The centers of cubes of two interlaced cubic grids (rescaled by factor 2), one composed of points with even and the other composed of points with odd Cartesian coordinates (in different colors). **b** BCC voxels centered at those points

A (binary) object O is a finite collection of cubes in the cubic grid. The cubes in O are called black. The cubes in the complement of O are called white. Connected components of O are the maximal connected subsets of O with respect to the chosen adjacency. An object O is called well-composed [26] if it has no critical vertices (incident with exactly two strictly vertex-adjacent black or exactly two strictly vertex-adjacent white cubes) and no critical edges (incident with exactly two strictly edge-adjacent black and exactly two strictly edge-adjacent white cubes).

For an object O, the associated cubical complex Q is a cell 3-complex that consists of the cubes in O together with all of their square faces, edges and vertices. An object O is well-composed if and only if $|Q|$ is a 3-manifold. The two definitions of well-composed 3D objects (discrete, through the critical configurations and continuous, through the notion of a manifold) are equivalent [2].

2.3 The BCC Grid

The body-centered cubic (BCC) [19] grid is a Voronoi tessellation of $\mathbb{R}^3$ associated with points in $(2\mathbb{Z})^3 \cup (2\mathbb{Z}+1)^3$, which can be seen as the centers of cubes of two interlaced cubic grids, rescaled by factor 2. The cubes of the first (second) grid are centered at points with even (odd) Cartesian coordinates (see Fig. 1a). The voxel of the BCC grid is a truncated octahedron, with eight regular hexagonal faces and six square faces, 36 edges and 24 vertices (see Fig. 1b).

An object O in the BCC grid is defined similarly to an object in the cubic grid: it is a finite set of (black) voxels in the grid. Its associated cell 3-complex Q is composed of all voxels in O and all their faces, edges and vertices.

The BCC grid has only one type of adjacency relation: if two voxels share a vertex or an edge, they also share a

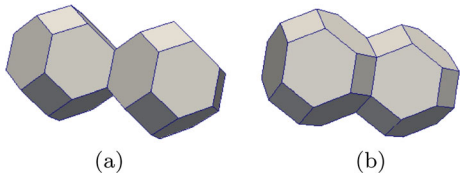


Fig. 2 Two BCC voxels which are not disjoint share an entire **a** square or **b** hexagonal face

face (see Fig. 2). Due to this property, there are no critical configurations in the BCC grid, and every object in this grid is well-composed. The shared face of two adjacent voxels may be hexagonal or square, and the two voxels are called hex-adjacent and quad-adjacent, respectively.

2.4 The Euler Characteristic

The Euler characteristic is a basic topological (homological) invariant, extensively used in many application domains, such as topological data analysis, image processing and pattern recognition and classification, to name just a few. It can be defined in several equivalent ways, and on different spaces, e.g., topological manifolds, cell complexes or polyhedra.

For a cell 3-complex Γ with c_0 vertices, c_1 edges, c_2 faces and c_3 3-cells (voxels), the Euler characteristic $\chi(\Gamma)$ is equal to

$$\chi(\Gamma) = c_0 - c_1 + c_2 - c_3 \tag{1}$$

For a general 3-complex Γ, this definition assumes that two cells are adjacent if they share at least one vertex. If applied to a cubical complex Q, it assumes vertex-adjacency for Q.

Alternatively, if Γ has β_0 connected components, β_1 tunnels and β_2 cavities, the Euler characteristic $\chi(\Gamma)$ is equal to $\beta_0 - \beta_1 + \beta_2$ [21]. A cavity is formally a bounded connected component of the background, and intuitively a hole inside the object, completely surrounded by it. A tunnel is intuitively a hole traversing the object from side to side, while a formal definition requires notions from homology theory. Intuitively, the number of tunnels is equal to the maximal number of non-separating cuts that can be made in the object.

In 3D, there is a connection between the Euler characteristic of a manifold 3-complex Γ and the Euler characteristic of its boundary $\partial\Gamma$ [5, 14, 28, 31], namely

$$\chi(\Gamma) = \chi(\partial\Gamma)/2 \tag{2}$$

and the Euler characteristic of $\partial\Gamma$ can be computed as

$$\chi(\partial\Gamma) = c_0 - c_1 + c_2 \tag{3}$$

where, here, c_0, c_1 *and* c_2 denote the number of vertices, edges and faces of $\partial\Gamma$.

2.5 Morse Theory

Morse theory studies the relationship between the topological shape of a manifold and (the critical points of) a function defined on the manifold, both in the continuous [30, 33] and in the piecewise linear [1, 11] setting.

For a C^2-differentiable real-valued function f defined over a closed compact manifold domain $M \subseteq \mathbb{R}^d$, a point $p \in M$ is a critical point of f if all first order partial derivatives of f (in a suitable coordinate system around p) vanish at p. The function f is a Morse function if the Hessian matrix $Hess_p f$ of the second derivatives of f at p is non-singular. Usually, it is supposed that there are no two critical points with the same function value. This property allows analyzing the topology changes at each critical point separately.

The number of negative eigenvalues of $Hess_p f$ is called the index of a critical point p. The corresponding eigenvectors point in the directions in which f is decreasing. A critical point p is a minimum or a maximum if it has index 0 or d, respectively. Otherwise, if the index of p is λ, $0 < \lambda < d$, p is a λ-saddle.

Let us consider a closed orientable 2-dimensional manifold M in $\mathbb{R}^3$, which is the boundary of a 3-manifold Γ, i.e., $M = \partial\Gamma$. On M, there are three types of critical points: minima, maxima and saddles. The topology (homotopy type) of the lower level sets $M_{\leq\alpha} = \{x \in M : f(x) \leq \alpha\}$ of f changes only at the critical points.

For example, let M be a (hollow) torus with horizontal axis of symmetry and let f be the vertical elevation. This is a 2-manifold without boundary and its Euler characteristic is 0. Figure 3 presents four stages of the evolution of the lower level set (shown in shaded color) while increasing the elevation. At the lowest point of M, the lower level set is created, as it initially a 2-manifold with boundary, homotopic to a disc ($\chi = 1$). At the lower saddle, it becomes homotopic to a cylinder ($\chi = 0$). At the higher saddle, two boundary circles of the cylinder touch to create a torus with a hole ($\chi = -1$). Until here the lower level set has been a 2-manifold with boundary. The highest point closes up the surface creating the internal cavity ($\chi = 0$ again).

In summary, minima create new connected components and maxima close off (create) cavities. Thus, each extremal critical point increases the Euler characteristic by 1. Saddles either merge two different connected components, or create a hole (a tunnel) by merging two parts of the same connected component. Thus, each saddle decreases the Euler characteristic by 1.

If m_λ denotes the number of critical points of f with index λ, then the Euler characteristic $\chi(M)$ of M is given by

$$\chi(M) = \sum_{\lambda=0}^{d} (-1)^\lambda m_\lambda,$$

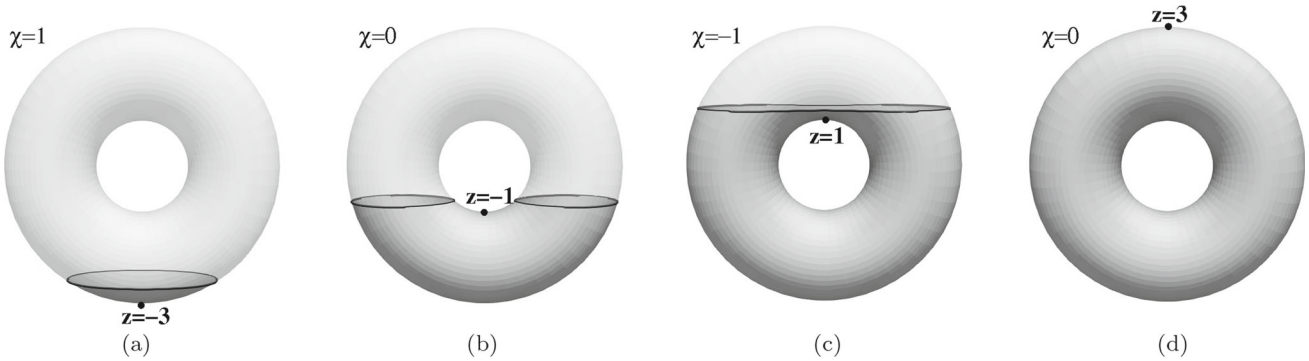


Fig. 3 Evolution of the lower level set of a torus surface. At each stage, the current lower level set is shaded. The lower level set is homeomorphic **a** to a disc, **b** to a cylinder, **c** to a torus with a hole, **d** to a torus

and, for a 2-manifold M,

$$\chi(M) = m_0 - m_1 + m_2$$

i.e., the number of minima, minus the number of saddles, plus the number of maxima.

2.6 The Discrete Gauss–Bonnet Theorem

The Gauss–Bonnet theorem gives a connection between the Gaussian curvature on a manifold M and its topology expressed through its Euler characteristic. It states that

$$\int_M K_G \, dA = 2\pi \chi(M),$$

where M is a closed 2-manifold without boundary, K_G is its Gaussian curvature and the integral is a surface integral over M [35].

The Descartes theorem gives a connection between the angles of the faces of a polyhedron P and its topology. It states that, if P is homeomorphic to the unit 2-sphere S^2, then

$$\sum_{v \in P} \delta(v) = 4\pi = 2\pi \chi(S^2),$$

where v is a vertex of P and the angular deficit $\delta(v)$ is the amount by which the sum of the face angles at v differs from 2π, i.e.,

$$\delta(v) = 2\pi - \sum_{v \in f} \alpha(v, f),$$

where f are the faces incident to the vertex v and $\alpha(v, f)$ is the internal angle of f at v.

The discrete version of the Gauss–Bonnet theorem extends the Descartes theorem to arbitrary (manifold) polyhedra. It states that

$$\sum_{v \in P} \delta(v) = 2\pi \chi(P),$$

where P is the given polyhedron [23] and v are the vertices of P. Intuitively, the Gaussian curvature in the interior of the faces and edges of P is equal to zero. The curvature is concentrated at the vertices of P and is equal to the vertex deficit. It follows that

$$\chi(P) = \frac{1}{2\pi} \sum_{v \in P} \delta(v) \tag{4}$$

2.7 Relation Between BCC and Cubic Grids

As common 3D acquisition devices provide data in a cubic grid, it is necessary to convert them to the BCC grid. This can be done in different ways.

A naive conversion can be performed in two ways: either we keep only cubes with all even or all odd coordinates and expand them to BCC voxels, or we insert a new datum at each cube vertex (with interpolated value) and place a BCC voxel at each vertex and each cube center. In the first case, we have half the resolution and we loose data, while in the second case we double the resolution. However there is no warranty that the Euler characteristic is preserved.

A method has been proposed in [7] which performs the conversion in the second way (i.e., by doubling the resolution), but choosing the color assigned to the cube vertices in such a way to preserve the Euler characteristic of the original cubic object according to vertex-adjacency or face-adjacency. Alternatively, Edelsbrunner and Kerber [12] have proposed a method transforming each cubic voxel into a (combinatorial) BCC voxel, by slightly shrinking it from one diagonal direction. The resolution is the same as the origi-

nal cubic grid, but in general the Euler characteristic is not preserved.

3 Related Work

A lot of algorithms have been proposed for the computation of the Euler characteristic of 3D objects in the cubic grid. The main drawback of this grid is that the value of the Euler characteristic depends on the adjacency model considered (classically vertex-adjacency or face-adjacency, but other adjacency models have been proposed as well [34, 37]), and it is unique only for 3-manifold objects. Here, we briefly review only the algorithms for computing the Euler characteristic in the cubic grid that are based either on the discrete version of the Gauss–Bonnet theorem, or on Morse theory.

The algorithm by Chen and Rong [6] works on a well-composed object O, with manifold boundary ∂O. Each vertex in ∂O is incident to three, four, five or six boundary faces. The sets of such vertices are denoted M_3, M_4, M_5 and M_6, and the numbers of such vertices are $|M_3|$, $|M_4|$, $|M_5|$ and $|M_6|$, respectively. Since each boundary face is a square, and each face angle is $\pi/2$, the deficit of each boundary vertex v incident to k boundary faces, $3 \leq k \leq 6$, is

$$\delta(v) = 2\pi - k\pi/2.$$

Thus, the vertices incident to four boundary faces do not affect the Euler characteristic, and the discrete Gauss–Bonnet theorem implies that

$$\chi(\partial O) = (|M_3| - |M_5| - 2|M_6|)/4.$$

The algorithm by Imiya and Eckhardt [18] makes a finer classification of the vertices in the boundary of a well-composed object O to obtain the same formula.

The algorithm by Lee at al. [27] works for objects with face-adjacency (or with vertex-adjacency, by considering the complement of the object), not necessarily well-composed. It is based on smoothing the black cubes (slightly inflating them and rounding the corners and edges) and applying the continuous Gauss–Bonnet theorem. It reduces to using a lookup table with vertex contributions to $\chi(O)$ for each possible configuration of $2 \times 2 \times 2$ cubes.

The algorithm by Čomić and Magillo [8] is based on counting only the boundary faces and vertices. The vertex count of the vertices where ∂O is non-manifold is adjusted depending on the chosen adjacency relation (face- or vertex-adjacency).

4 Algorithms for the Computation of the Euler Characteristic in the BCC Grid

We describe five algorithms to compute the Euler characteristic of an object in the BCC grid. The first two algorithms are the classic ones based on cell counting. The other three algorithms are new: one is based on Morse theory and critical points, and two algorithms are based on summing angles.

Our implementation of all algorithms uses the 4-valued coordinate system proposed in [7]. In it, every voxel, face, edge and vertex in the BCC grid is represented by four dependent coordinates, whose sum is zero. Of the four coordinates of a BCC voxel centered in the first interlaced cubic grid, two belong to $\mathbb{Z}_8$ and two to $\mathbb{Z}_8 + 4$; for a BCC voxel centered in the second interlaced cubic grid, two coordinates belong to $\mathbb{Z}_8 + 2$ and two to $\mathbb{Z}_8 + 6$. The coordinates of a face are the average of the coordinates of its two incident BCC voxels. The coordinates of an edge (shared by two hex- and one quad-face) are the average of the coordinates of its two incident hex-faces. The coordinates of a vertex are the average of the coordinates of its four incident BCC voxels. All adjacency and incidence relations are retrieved by arithmetic operations on the four coordinates of the involved cells.

The input object O is given as a list of (black) BCC voxels. In addition, each voxel of (the portion of) the BCC grid containing O is marked as black or white. In this setting, in order to find, for example, the black voxels adjacent to a given voxel x, we access the 14 neighbors of x and test their color in constant time. The surface ∂O, needed by the boundary-based algorithms, is represented as a list of faces.

4.1 Algorithm Based on Counting Cells

The straightforward classic approach simply computes the alternating sum of the number of i-cells. It admits a volume-based version, which computes the Euler characteristic of a three-dimensional object O in the BCC grid with Formula (1), and a surface-based version, that does it by computing the Euler characteristic of the boundary surface ∂O of O with Formula (3).

4.1.1 Volume-Based Version (VOL)

This algorithm that we denote as **VOL** counts each black voxel and its faces, edges and vertices, paying attention to count a face, edge or vertex in just one of the voxels containing it.

Our implementation scans the black voxels and processes each one. Processing of a voxel x consists of the following actions:

- count x and mark it;
- access the 14 adjacent voxels of x, and check the configuration. Based on the following quantities:
  - N_4 and N_6, the number of quad- and hex-faces, respectively, of x, whose other incident voxel is black and already marked; we call them black faces;

  - $N_{6,6}$, the number of edges of x, shared by two hex-faces, such that their two incident voxels, different from x, are black and already marked;
  - $N_{4,6}$, the number edges of x, shared by a quad- and a hex-face, such that their two incident voxels, different from x, are black and already marked;
  - $N_{4,6,6}$, the number of vertices of x (each shared by one quad- and two hex-faces of x) such that their three incident voxels, different from x, are black and already marked;
- count $14 - N_4 - N_6$ faces for x (from the 14 faces of x, subtract the ones already counted in the adjacent black voxels);
- count $36 - 4N_4 - 6N_6 + N_{6,6} + N_{4,6}$ edges for x (from the 36 edges of x, subtract the ones belonging to black faces to avoid counting them twice, but those edges shared by two black faces must not be subtracted twice),
- count $24 - 4N_4 - 6N_6 + 2(N_{6,6} + N_{4,6}) - N_{4,6,6}$ vertices for x (from the 24 vertices of x, subtract the ones belonging to black faces, with similar considerations as for edges, but a vertex may be shared by up to three black faces).

4.1.2 Surface-Based Version (SUR)

This version, that we denote as **SUR** assumes that the boundary surface of the object (consisting of hexagonal and square faces) is available. The algorithm performs a loop on input boundary faces. For each face f, it counts the face f itself and potentially counts each edge and vertex of f. Suitable attention is payed to count each edge or vertex only at one of the faces it belongs to. For such purpose, in our implementation we consider the lexicographic (total) order of the face coordinates. An edge e is counted only if the current boundary face precedes the other boundary face incident in e, and a vertex v is counted only if the current boundary face is minimum among the boundary faces incident in v.

Thanks to the fact that ∂O is 2-manifold (each edge belongs to two faces), we could count $1/2$ for each edge of a face. Nevertheless, a mechanism such as the total order is still necessary to count the vertices correctly.

4.2 Algorithm Based on Morse Theory (MOR)

The new algorithm based on Morse theory, that we denote as **MOR**, computes the Euler characteristic of the boundary ∂O by recording the changes in the lower level sets with a suitable elevation function f, defined in such a way that no two vertices have the same function value (as unicity is a usual requirement for Morse functions, see Sect. 2.5).

This is done by considering the lexicographic order of the (z, y, x) triplets obtained from the coordinates of all the vertices, and function f maps each vertex onto its "elevation" defined as its position in the sorted list of all vertices. Intuitively, this is equivalent to sweeping ∂O through a set of parallel almost horizontal planes moving in the direction of the positive z axis. Such planes are tilted slightly so that the plane encounters one vertex at a time among the ones with a given z. The change of the Euler characteristic when sweeping over a vertex v depends on the relative position (elevation) of the neighbors of v. Each vertex v in ∂O is adjacent to either three or four other vertices in ∂O, and it is processed as follows:

- If all neighbors come after v (they are higher than v w.r.t. the elevation function f), then v creates a new component of the lower level set, and the Euler characteristic is increased by 1.
- If all neighbors come before v (they are lower than v w.r.t. f), then v closes off a cavity in the lower level set, and the Euler characteristic is also increased by 1.
- If some neighbors come before v and some after v (some are higher and some are lower than v w.r.t. f), and the ones coming before (after) v are consecutive around v, then v is a regular vertex inducing no change in the Euler characteristic of the lower level set.
- If v has four neighbors in ∂O, which are alternating before-after-before-after v w.r.t. f, then v either merges two different connected components, or it creates a tunnel by connecting two pieces of the same connected component. The Euler characteristic of the lower level set is decreased by 1.

The justification for such increments is given in Sect. 2.5.

The implementation first loops on the faces of the given object boundary ∂O, extracts the vertices of each hex-face (this guarantees to get each vertex at least once) and puts them into an array. Then, the array is sorted lexicographically and duplicates are removed.

Before starting the main loop on the sorted vertices, $\chi = 0$ is set. Then, a sweep over the vertices of ∂O, contained in the sorted array, is performed. These are the actions performed while sweeping over a vertex v:

- Get the (at most four) boundary faces incident in v;
- For each incident boundary face f, get the vertices w_1, w_2 preceding and following v along the boundary of f. If $w_1 < v < w_2$ or $w_1 > v > w_2$ in the lexicographic order, then record one switch;
- Let s be the total number of recorded switches at all incident boundary faces of v (s is even and $0 \leq s \leq 4$). If $s = 0$ then increment χ; if $s > 2$ then decrement χ; otherwise, leave χ unchanged.

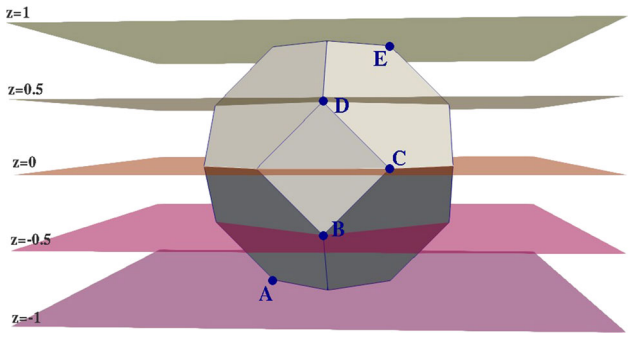


Fig. 4 The object composed of one voxel is swept by moving a plane which stops at the vertices

Figure 4 shows how the algorithm works on a small input consisting of just one voxel. Here, each vertex has three neighbors, so it cannot have more than two sign switches. Initially $\chi = 0$. For the first vertex with $z = -1$, e.g., A, all neighbors are not yet swept, there are no switches and χ increases to 1. The other vertices with $z = 0$ have one or two swept neighbors, therefore two switches and χ is not changed. The same happens for all other vertices, with the exception of the last swept vertex with $z = 1$. For the last one, e.g., E, all neighbors are already swept, there are no switches and χ increases to 2. Therefore, $\chi(O) = \chi(\partial O)/2 = 1$.

The type of the change in χ induced by each vertex v is determined locally, as it depends only on the neighbors of v in ∂O and their relative position with respect to v.

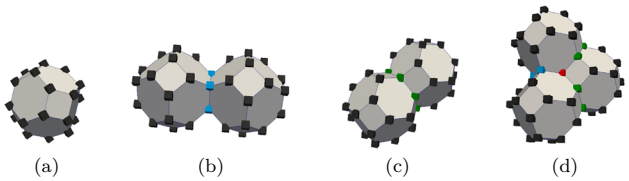
4.3 First Algorithm Based on the Discrete Gauss–Bonnet theorem (GB1)

This algorithm implements Formula (4): it computes and sums the contributions $\kappa(v) = \delta(v)/2\pi$ of each vertex v, i.e., its angle deficit divided by 2π. In order to determine the contributions, we classify the vertices on the boundary ∂O of O based on the local configuration of the incident voxels, into four classes:

1. M_1 is the set of vertices incident to one voxel in O,
2. M_2^q is the set of vertices incident to two quad-adjacent voxels in O (e.g., the four blue vertices in Fig. 5b),
3. M_2^h is the set of vertices incident to two hex-adjacent voxels in O (e.g., the six green vertices in Fig. 5c),
4. M_3 is the set of vertices incident to three voxels in O (e.g., the red vertex in Fig. 5d).

We compute the contribution κ of each vertex in ∂O to the Euler characteristic $\chi(\partial O)$ as follows:

1. A vertex $v \in M_1$ is incident to one quad-face and two hex-faces, so
   $\kappa(v) = \frac{1}{2\pi} \cdot (2\pi - (2 \cdot \frac{2\pi}{3} + \frac{\pi}{2})) = \frac{1}{12}$.



Config.	M_1	M_2^q	M_2^h	M_3	$\chi(\partial O)$
(a)	24	0	0	0	$\frac{1}{12}24 = 2$
(b)	40	4	0	0	$\frac{1}{12}40 - \frac{1}{3}4 = 2$
(c)	36	0	6	0	$\frac{1}{12}36 - \frac{1}{6}6 = 2$
(d)	46	2	8	2	$\frac{1}{12}46 - \frac{1}{3}2 - \frac{1}{6}8 + \frac{1}{12}2 = 2$

Fig. 5 An object O composed of **a** a single voxel, **b** two quad-adjacent voxels, **c** two hex-adjacent voxels, **d** two quad-adjacent voxels and a third voxel hex-adjacent to both. Vertices belonging to M_1, M_2^q, M_2^h and M_3 are depicted in black, blue, green and red, respectively. The table shows the number of vertices in each set and the computation of the Euler characteristic of ∂O

2. A vertex $v \in M_2^q$ is incident to four hex-faces, so
   $\kappa(v) = \frac{1}{2\pi} \cdot (2\pi - (4 \cdot \frac{2\pi}{3})) = -\frac{1}{3}$.
3. A vertex $v \in M_2^h$ is incident to two quad-faces and two hex-faces, so
   $\kappa(v) = \frac{1}{2\pi} \cdot (2\pi - (2 \cdot \frac{2\pi}{3} + 2 \cdot \frac{\pi}{2})) = -\frac{1}{6}$.
4. A vertex $v \in M_3$ is incident to one quad-face and two hex-faces, so
   $\kappa(v) = \frac{1}{2\pi} \cdot (2\pi - (2 \cdot \frac{2\pi}{3} + \frac{\pi}{2})) = \frac{1}{12}$.

Then, $\chi(\partial O) = \sum_{v \in \partial O} \kappa(v)$.

In Fig. 5, we show the computation of the Euler characteristic according to the above formula on some examples, all having the Euler characteristic $\chi(\partial O)$ equal to 2.

The algorithm that we denote as **GB1** visits all vertices in ∂O, classifies them according to the M set they belong to, and iteratively adds the corresponding contribution of the vertex to χ.

For convenience, the implementation computes the quantity 12χ and, at the end, divides it by 12. Initially, 12χ is set to zero. Then, the algorithm loops on the given faces of ∂O. For each face f, it gets its vertices and, for each vertex v of f, if v is not marked, it computes the contribution of v and marks it. Vertex marking ensures that each vertex is considered only once.

4.4 Second Algorithm Based on the Discrete Gauss–Bonnet Theorem (GB2)

The second algorithm that we denote as **GB2** also refers to Formula (4), but uses the fact that the sum of the angles over the vertices in ∂O is equal to the sum of the angles over the faces in ∂O. For quad-faces, this sum is 2π, and for hex

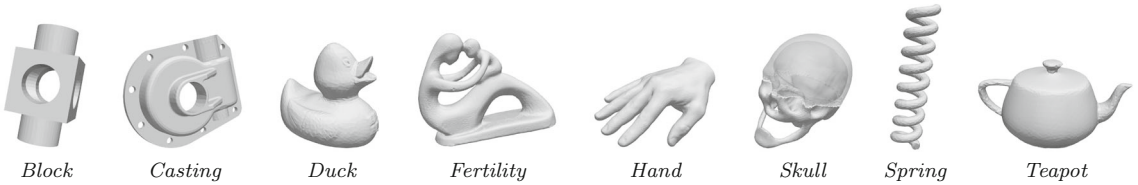


Fig. 6 Shapes whose BCC discretizations have been used in the experiments. In addition, a solid ball *Ball* and a hollow ball *Bubble* have been used

Table 1 Information about the discretized shapes composing the first test set used in the experiments. N denotes the side of the reference grid, n_O the number of black voxels, $n_{\partial O}$ the number of boundary faces, $r = n_{\partial O}/n_O$, and $r \times N$

Object	N	n_O	r	$r \times N$	Object	N	n_O	r	$r \times N$
Block	50	33,742	1.139	56.95	Hand	50	16,870	1.342	67.10
	100	269,430	0.569	56.90		100	134,917	0.676	67.60
	150	911,502	0.380	57.00		150	455,364	0.451	67.65
	200	2,160,012	0.287	57.40		200	1,079,177	0.338	67.60
	250	4,217,120	0.229	57.25		250	2,103,947	0.271	67.75
Casting	50	8712	4.011	200.55	Skull	50	31,021	2.089	104.45
	100	69,197	2.052	205.20		100	247,644	1.049	104.90
	150	229,415	1.400	210.00		150	835,959	0.700	105.00
	200	545,201	1.050	210.00		200	1,981,223	0.525	105.00
	250	1,069,139	0.837	209.25		250	3,870,233	0.420	105.00
Duck	50	63,287	0.672	33.60	Spring	50	2944	4.213	210.65
	100	506,274	0.338	33.80		100	23,510	2.129	212.90
	150	1,708,533	0.225	33.75		150	79,307	1.425	213.75
	200	4,049,806	0.169	33.80		200	188,129	1.070	214.00
	250	7,909,964	0.135	33.75		250	367,342	0.857	214.25
Fertility	50	13,631	1.763	88.15	Teapot	50	27,117	0.760	38.00
	100	109,258	0.886	88.60		100	216,884	0.383	38.30
	150	368,651	0.593	88.95		150	835,959	0.224	33.60
	200	874,053	0.445	89.00		200	1,734,889	0.192	38.40
	250	1,707,078	0.356	89.00		250	3,388,194	0.153	38.25

faces, it is 4π. Then,

$$\begin{aligned} 2\pi\chi(\partial O) &= \sum_{v\in\partial O} \delta(v) \\ &= \sum_{v\in\partial O} (2\pi - \sum_{v\in f} \alpha(v, f)) \\ &= \sum_{v\in\partial O} 2\pi - \sum_{v\in\partial O} \sum_{v\in f} \alpha(v, f) \\ &= 2\pi c_0 - \sum_{f\in\partial O, v\in f} \alpha(v, f) \\ &= 2\pi c_0 - 2\pi c_2^q - 4\pi c_2^h. \end{aligned}$$

The same formula could be obtained from the known relation $\chi(\partial O) = c_0 - c_1 + c_2 = c_0 - c_1 + c_2^q + c_2^h$ by noting that each edge is shared by two faces, because ∂O is manifold. Therefore, $2c_1 = 4c_2^q + 6c_2^h$ and $\chi(\partial O) = c_0 - c_2^q - 2c_2^h$.

The algorithm visits all faces in ∂O and counts the number c_2^q of quad-faces and the number c_2^h of hex-faces. It also counts the number c_0 of vertices in ∂O, by counting, for each face, all its vertices which are not yet marked, and then marking them. Then, the Euler characteristic is computed as

$$\chi(\partial O) = c_0 - c_2^q - 2c_2^h.$$

For example, in Fig. 5b we have 44 vertices, 10 quad-faces and 16 hex-faces; therefore, $\chi(\partial O) = 44 - 10 - 32 = 2$.

4.5 Computational Complexity

We evaluate the time complexity of the algorithms in the worst case. We denote with n_O the number of voxels in the object O and with $n_{\partial O}$ the number of faces in its boundary surface ∂O.

Algorithm **VOL** is in $O(n_O)$ and **SUR** is in $O(n_{\partial O})$, because they perform a constant amount of work for each voxel in O, and for each face in ∂O, respectively. **SUR** does not need to compute the sorted sequence of faces in the lexicographic order as it just tests whether one face comes before or after another (adjacent) one.

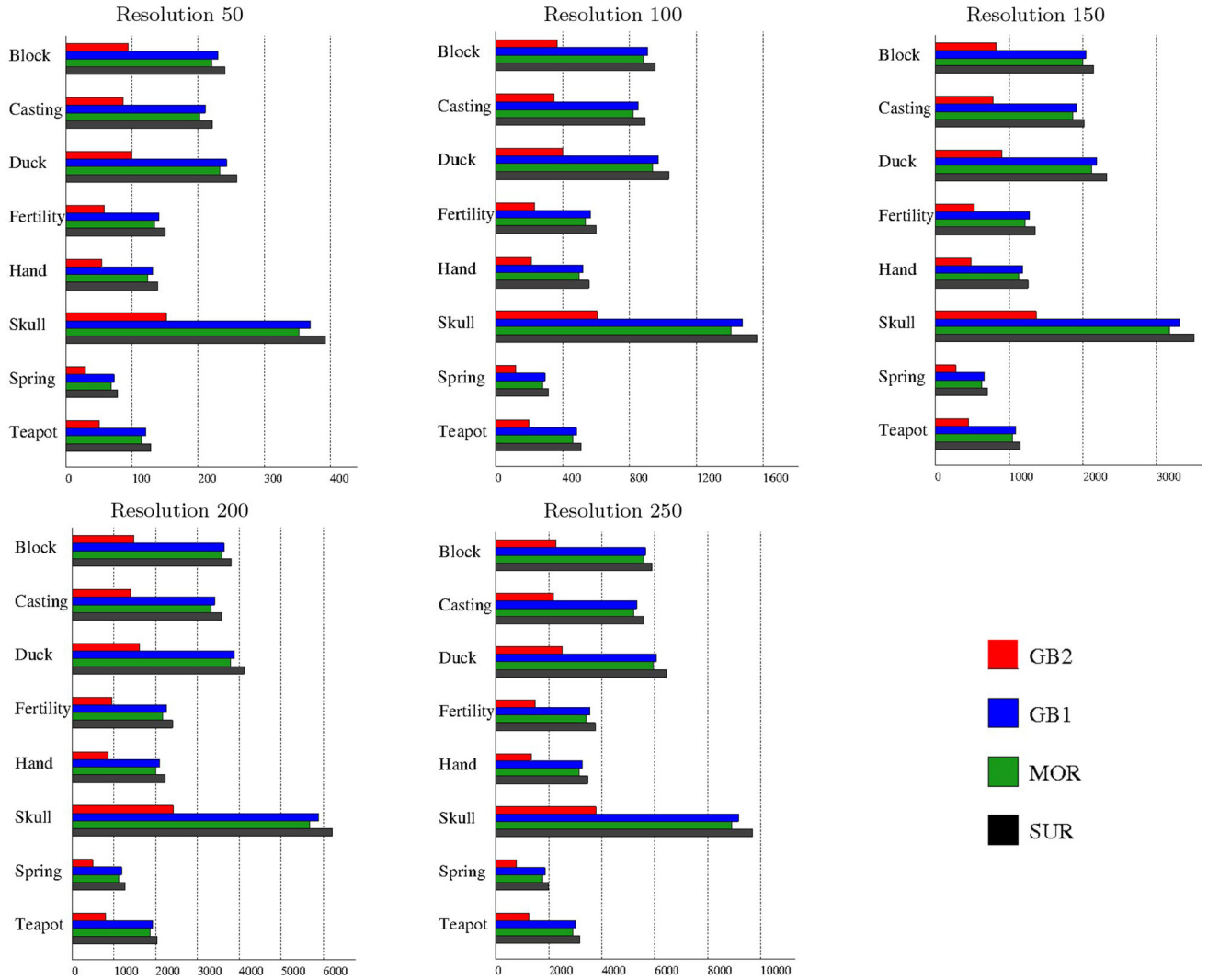


Fig. 7 Running times of the boundary-based algorithms **SUR**, **MOR**, **GB1** and **GB2**. Times are in milliseconds

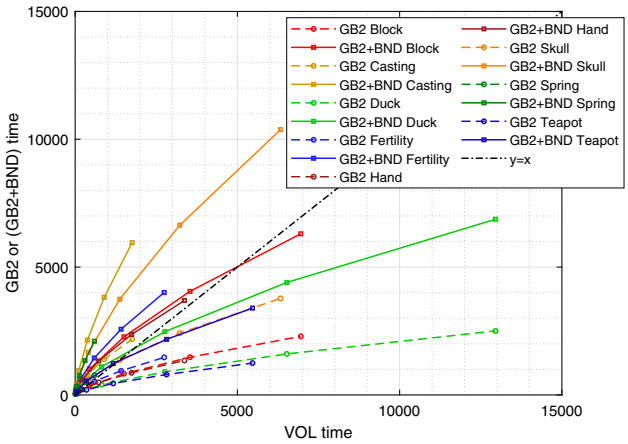


Fig. 8 Running times of **GB2** and of **GB2+BND** (on y-axis) compared to the running time of **VOL** (on x-axis). The dots corresponding to the same test objects at different resolutions are connected by segments (resolution increases from left to right)

Algorithm **MOR** is in $O(n_{\partial O} \log(n_{\partial O}))$ due to the preliminary sorting of the boundary vertices in the lexicographic order, which dominates overall time complexity.

Algorithms **GB1** and **GB2** are in $O(n_{\partial O})$. For example, **GB1** scans the faces of ∂O, gets the (four or six) vertices of each face, for each vertex v gets the incident faces of v in the BCC grid (they are four hex-faces and two quad-faces) and counts the ones belonging to ∂O (this is done in constant time because we mark the faces of ∂O) to compute the contribution $\kappa(v)$. In total, **GB1** visits $\leq 6n_{\partial O}$ vertices and their $\leq 36n_{\partial O}$ incident faces, that is $O(n_{\partial O})$.

5 Experiments and Results

We implemented all the algorithms for computing the Euler characteristic of an object in the BCC grid, described in

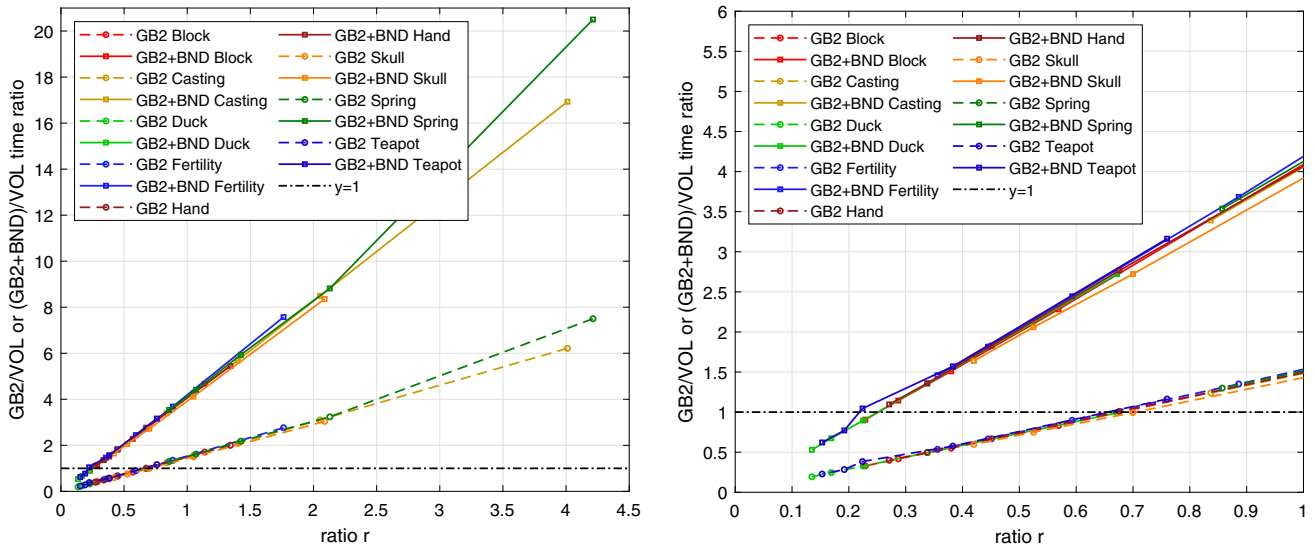


Fig. 9 The ratios of running times of **GB2** and of **GB2+BND** over the running time of **VOL** as functions of $r = n_{\partial O}/n_O$. The right image is a detail of the left one, focusing on $r < 1$

Sect. 4, namely algorithms **VOL**, **SUR**, **MOR**, **GB1** and **GB2**.

We also implemented an algorithm **BND** to extract the boundary surface of a given BCC object, which is needed to produce the input for surface-based algorithms to compute the Euler characteristic (i.e., all the above ones, but **VOL**).

The algorithms have been coded in C language and run on a PC equipped with an Intel CPU i7-2600K CPU at 3.4 Gigahertz with 32 Gigabytes of RAM.

Our experiments are aimed at:

1. Comparing the four surface-based algorithms among them, identifying the most efficient one.
2. Comparing the surface-based and the volume-based approach to the Euler characteristic computation, identifying the conditions under which the former or the latter is more efficient.

5.1 Comparing the Four Boundary-Based Algorithms

Our test set is obtained by discretizing some shapes from the Digital Shape Workbench,[1] shown in Fig. 6. Such shapes are given as tetrahedral meshes filling an object, or as triangle meshes bounding it. All the shapes have been discretized by first rescaling them so that their bounding box fits a cubic grid of side N (i.e., N^3 cubes) and then considering the BCC grid, where a BCC voxel is centered at each cube center and at each cube vertex. A BCC voxel has been set to black if and only if its center lies inside the object. The used grid sides are $N = 50, 100, 150, 200, 250$.

[1] http://visionair.ge.imati.cnr.it/ontologies/shapes/.

Table 1 summarizes the information about the resulting BCC images: the object name, the grid side N, the number n_O of black voxels, and the ratio $r = n_{\partial O}/n_O$ (where $n_{\partial O}$ is the number of boundary faces). As the volume of an object grows roughly with N^3 and the area of its boundary surface with N^2, the ratio r decreases with resolution and is inversely dependent on the grid side N. The last column in Table 1 shows the value of $r \times N$, which is roughly constant for each shape.

The running times, in milliseconds, are shown in Fig. 7. As expected, the running times are linearly dependent on the size $n_{\partial O}$ of the boundary surfaces, with a constant that is roughly 0.0062 for **SUR**, 0.0056 for **MOR**, 0.0059 for **GB1** and 0.0024 for **GB2** (with variations involving the fourth decimal digit). Therefore, each of the three new boundary-based algorithms represents an improvement over the classic one **SUR** based on cell counting.

The fastest method is **GB2**, i.e., the second algorithm using the Gauss–Bonnet theorem. The running time of **GB2** is less than 39% of **SUR**, less than 44% of **MOR**, and less than 42% of **GB1**. Algorithms **MOR** and **GB1** are not much faster than **SUR**, with the former slightly faster than the latter (95% and 96%, respectively).

5.2 Comparison with the Volume-Based Approach

As expected, the running times of **VOL** are linearly dependent on the number n_O of black voxels, with a constant roughly equal to 0.0016 (with variations involving the fifth decimal digit).

We compare the fastest boundary-based algorithm, i.e., **GB2**, with the volume-based one **VOL**, with the aim of

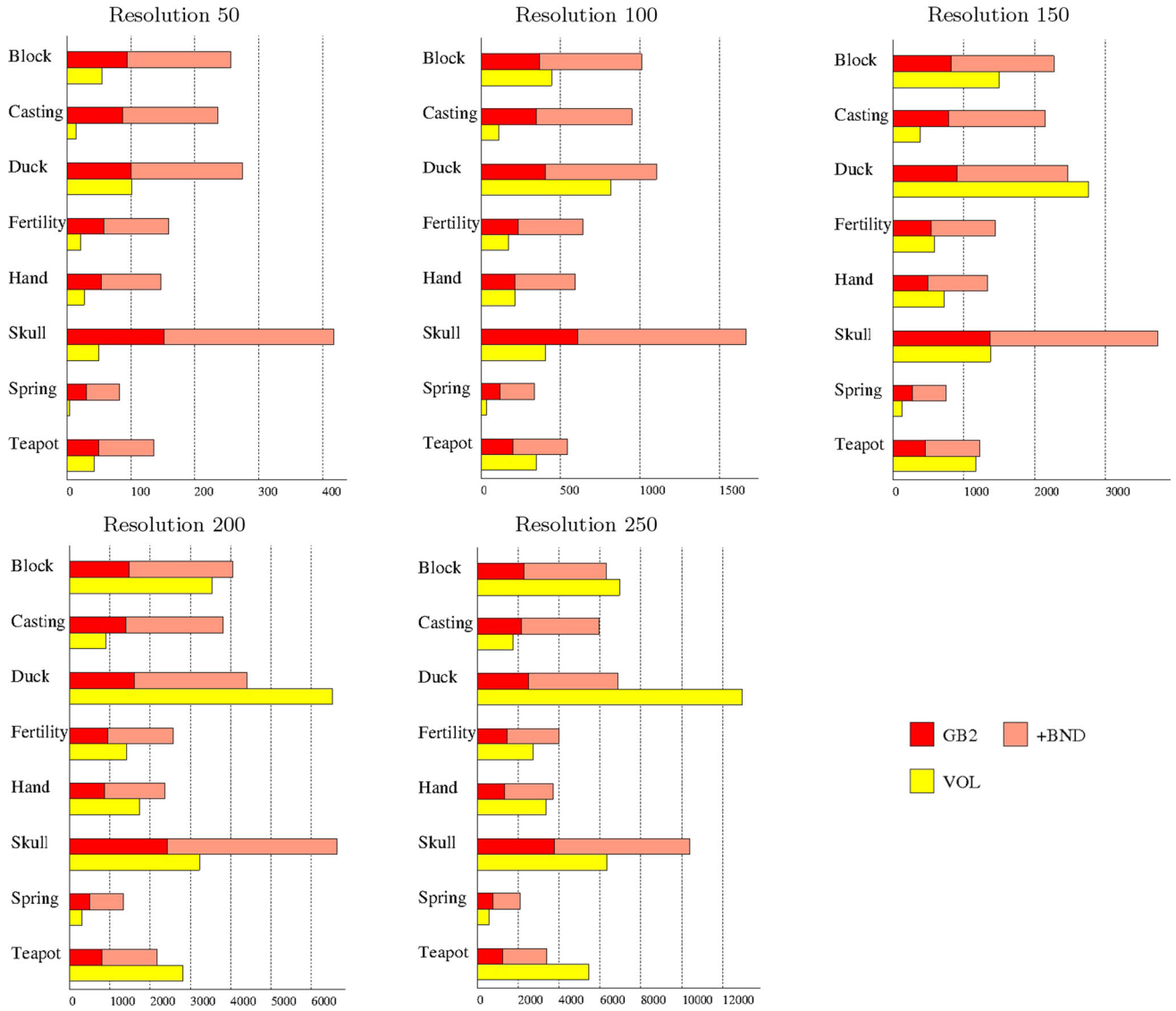


Fig. 10 Running times of the fastest boundary-based algorithm **GB2** (red), the algorithm for extracting the boundary surface (orange), and the volume-based algorithm for Euler computation **VOL** (yellow). Times are in milliseconds

understanding under which conditions the former is convenient over the latter. If the boundary surface is either given, or has to be extracted for other purposes, it is sufficient that the running time of **GB2** is smaller than the running time of **VOL**. Otherwise, it is necessary that the sum of execution times of **GB2** and of boundary extraction **BND** is smaller than **VOL**. Figure 10 shows the above running times, obtained on the same test set used in Sect. 5.1. Referring to Fig. 10, we compare the bottom yellow rectangle (**VOL**) with either the red rectangle (**GB2**) or the union of the red and orange rectangles (**GB2+BND**), i.e., the left part of the top rectangle, or the entire top rectangle.

For an easier comparison, Fig. 8 plots the running times of **VOL** on the x-axis and of **GB2** and **GB2+BND** on the y-axis. A line connects the five resolutions of each test object. The cases in which **GB2** (or **GB2+BND**) is faster than **VOL** are the dots lying below the line $y = x$.

The boundary-based approach to Euler computation is more convenient than the volume-based one when the boundary surface is small compared to its enclosed volume (in terms of the number of cells), i.e., when the ratio $r = n_{\partial O}/n_O$ is small. The plots in Fig. 9 show the ratio of running times **GB2/VOL** and (**GB2+BND**)**/VOL** as a function of r. Indeed, Fig. 9 shows a linear dependency on r, with factor $\simeq 1.5$ for **GB2/VOL** and $\simeq 4.1$ for (**GB2+BND**)**/VOL**. **GB2** is faster than **VOL** when $r < 0.66$. The total running time **GB2+BND**, including boundary extraction, is smaller than that of **VOL** when $r < 0.25$.

Small values of r are more likely to happen when the shape is "fat" and when the resolution of the discretization (the

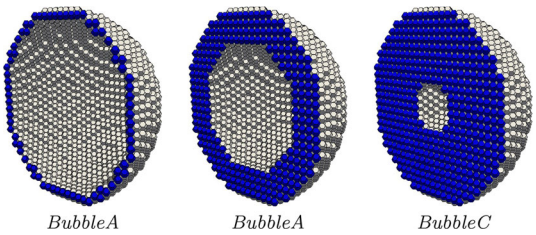


Fig. 11 Hollow spheres with three different widths (the inner radius is 92%, 68%, 36% of the outer radius, respectively). The shapes are shown with reference cubic grid side $N = 25$

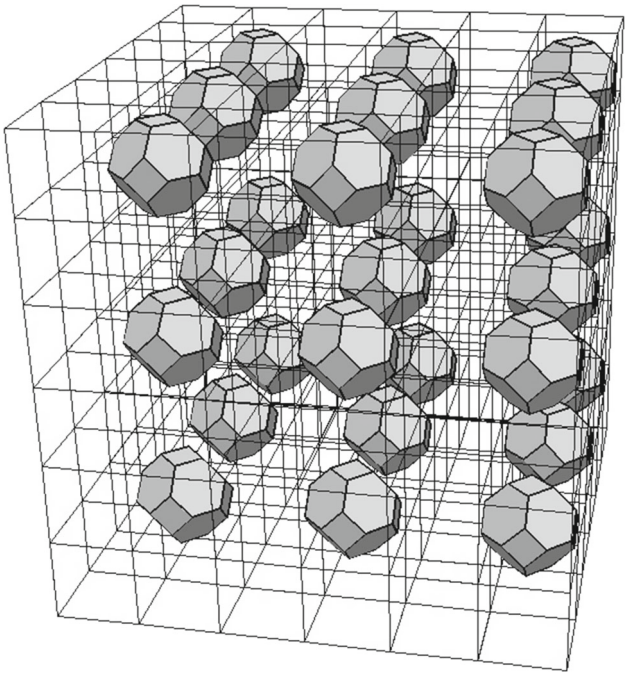


Fig. 12 Configuration of M^3 isolated BCC voxels, with $M = 3$

size N of the grid) is high. Fat objects (e.g., *Duck* or *Teapot*) have $r < 0.66$ for (almost) all values of N and $r < 0.25$ for $N \geq 150$. Objects with thin or elongated parts, such as *Casting* and *Spring*, never have $r < 0.66$ in our experiments. In Sect. 5.3 we will analyze the relation between the shape of an object and the ratio r, influencing the relative performance of the algorithms.

5.3 Analysis of the Impact of the Object Shape

In Sect. 5.2 we noted that the boundary-based algorithm **GB2** outperforms the volume-based one **VOL** when the ratio $r = n_{\partial O}/n_O$ is small enough. To further analyze the relation between the shape of an object, the resolution of the discretization, and the ratio r, we built three artificial test sets:

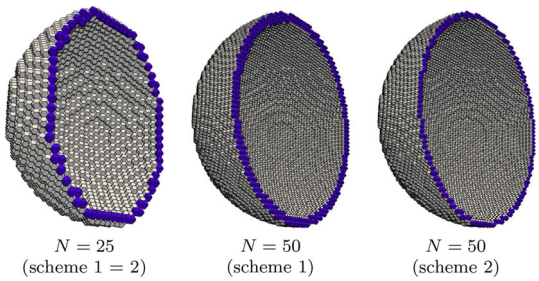


Fig. 13 The two different discretization schemes for hollow spheres *BubbleA* and *BubbleA'*. At the lower resolution $N = 25$ the width is one voxel and 8% of the outer radius. At double resolution $N = 50$, the width is two voxels and still 8% of the outer radius in the first scheme. In the second scheme, the width remains one voxel and becomes 4% of the outer radius

1. A set of disjoint BCC voxels. This is the object with the largest boundary surface with respect to its volume. The ratio is $r = 14$ because all faces of each voxel lie on the boundary. Our isolated voxels are arranged in a regular pattern. Considering a reference cubic grid of side N, they are located at the centers of the cubes with even position in all three coordinate axes. So, the number of black BCC voxels is M^3 with $M = N/2$. The test objects, called *Voxels*, have been generated for $N = 10, 20, 40, 60, 80, 100, 120, 140, 160$. Figure 12 shows a smaller example, with $N = 6$ and $M = 3$.
2. Four (hollow) spheres with different width: 8%, 32%, 64% and 100% of the outer radius. We denote such objects as *BubbleA, BubbleB, BubbleC*, and *Ball*, respectively. The last one is a solid sphere with no cavity, and it is the object with the smallest boundary surface with respect to its volume. The other three hollow spheres correspond to a decreasing ratio r and are shown in Fig. 11. Such spheres have been discretized, as done for the shapes in Sect. 5.1, with reference grid of side $N = 25, 50, 75, 100, 125, 150, 200$. The width of the hollow spheres remains a fixed percentage of the outer radius and spans a larger number of voxels at higher resolution.
3. Another three hollow spheres, called *BubbleA', BubbleB', BubbleC'*, with the same width as the above ones at the base resolution ($N = 25$), but using another discretization scheme at higher resolutions: the width (difference between outer and inner radius) remains constant in terms of the number of voxels, at all resolution. The cavity gets larger at higher resolutions. Figure 13 illustrates the two discretization schemes.

In *Voxels*, the number $n_{\partial O}$ of boundary faces is 14 times the number n_O of black voxels, so the running times of both **GB2** and **VOL** are linear in n_O. The ratio of running times **GB2/VOL** is slightly decreasing as n_O grows (from 23 with $N = 10$ and 125 voxels, to 21 with $N = 160$ and

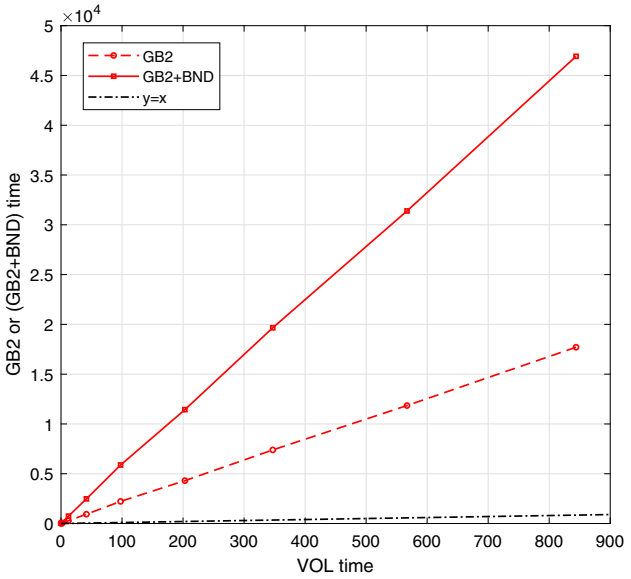


Fig. 14 Isolated voxels. The ratios of running times of **GB2** and of **GB2+BND** as a function of the running time of **VOL**

Table 2 Information about the BCC objects representing solid and hollow spheres. N denotes the side of the reference grid, n_O the number of black voxels, $n_{\partial O}$ the number of boundary faces, $r = n_{\partial O}/n_O$, and (for the first discretization scheme) $r \times N$

	N	n_O	r	$r \times N$
Ball	25	16,361	0.771	19.274
	50	131,019	0.389	19.441
	75	441,711	0.259	19.400
	100	1,047,289	0.194	19.384
	125	2,045,557	0.155	19.398
	150	3,535,559	0.129	19.395
	175	5,612,135	0.111	19.391
	200	8,376,753	0.097	19.395
	N	n_O	r	$r \times N$
BubbleA	25	3686	6.336	158.410
	50	29,182	3.220	160.997
	75	97,682	2.158	161.849
	100	231,938	1.616	161.641
	125	452,886	1.294	161.757
	150	783,734	1.077	161.528
	175	1,242,630	0.924	161.663
	200	1,853,562	0.809	161.859
BubbleB	25	11,296	1.624	40.590
	50	89,970	0.826	41.309
	75	302,848	0.552	41.392
	100	717,906	0.414	41.368
	125	1,402,624	0.331	41.372
	150	2,424,482	0.276	41.353
	175	3,847,716	0.236	41.361
	200	5,743,214	0.207	41.367

Table 2 continued

	N	n_O	r	$r \times N$
BubbleC	25	16,022	0.844	21.090
	50	128,182	0.428	21.402
	75	431,990	0.285	21.372
	100	1,024,294	0.214	21.372
	125	2,000,582	0.171	21.387
	150	3,457,894	0.146	21.385
	175	5,489,138	0.122	21.376
	200	8,192,746	0.107	21.385
	N	n_O	r	
BubbleA'	25	3686	6.336	
	50	15,416	6.319	
	75	34,370	6.472	
	100	62,144	6.406	
	125	97,490	6.404	
	150	140,840	6.404	
	175	189,862	6.477	
	200	248,468	6.473	
BubbleB'	25	11,296	1.624	
	50	53,354	1.627	
	75	126,916	1.618	
	100	231,938	1.616	
	125	368,332	1.616	
	150	537,898	1.611	
	175	735,448	1.616	
	200	965,530	1.616	
BubbleC'	25	16,022	0.844	
	50	96,692	0.742	
	75	247,942	0.727	
	100	470,240	0.722	
	125	762,898	0.721	
	150	1,127,096	0.720	
	175	1,560,886	0.719	
	200	2,064,188	0.719	

512K voxels). This probably depends on the hidden constants, and on the larger amount of fixed work in **VOL**, that becomes relevant for small n_O. The ratio of running times (**GB2+BND**)/**VOL** ranges from 55 to 60.5, with similar, but more noisy, variations when increasing N. These behaviors are plotted in Fig. 14.

Table 2 summarizes the information for the (hollow) spheres with the two discretization schemes, showing the same information as Table 1.

For *Ball* and *BubbleA, BubbleB, BubbleC*, the behavior of r is similar to the shapes in Table 1, i.e., r is roughly inversely proportional to the grid side N. For *BubbleA', BubbleB',*

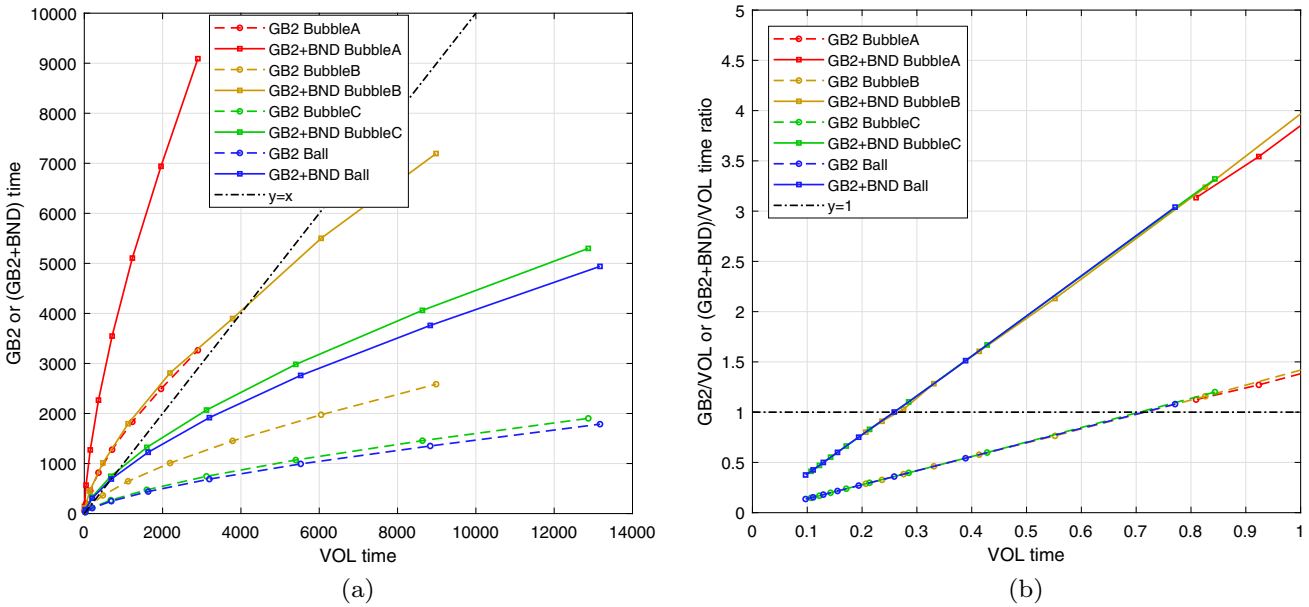


Fig. 15 Running times for the solid sphere *Ball* and the three hollow spheres *BubbleA,B,C* with the first discretization scheme. **a** The ratios of running times of **GB2** and of **GB2+BND** as a function of the running time of **VOL**. **b** The ratios of running times of **GB2** and of **GB2+BND** over the running time of **VOL** as functions of $r = n_{\partial O}/n_O$ (detail for $r < 1$)

BubbleC', instead, r is almost constant w.r.t. the resolution N.

The running times of **VOL** confirm to be linear in the number n_O of black voxels, with a multiplicative factor $\simeq 0.0016$. The running times of **GB2** are roughly linear in the number $n_{\partial O}$ of boundary faces, with a factor that is slightly smaller for *Ball, BubbleA, BubbleB, BubbleC* (0.0021–0.0022) than for *BubbleA', BubbleB', BubbleC'* (0.0023–0.0025).

Confirming the results in Sect. 5.2, **GB2** is faster than **VOL** when $r < 0.7$ and **GB2+BND** is faster than **VOL** when $r < 0.25$. The plots in Figs. 15 and 16 compare the running times of **GB2** and **GB2+BND**, with the running times of **VOL**.

In *Ball*, $r = 0.77$ at the lowest resolution, and at higher resolutions it ranges from 0.39 to 0.15; therefore, **GB2** is almost always the best algorithm. In the hollow spheres *BubbleA, BubbleB, BubbleC*, r becomes < 0.7 at a resolution that is coarser if the inner cavity is smaller. This confirms that the boundary-based approach is preferable for fat objects.

In *BubbleA', BubbleB', BubbleC'*, the ratio r is almost constant w.r.t. the resolution, and so is the ratio of running times of **GB2** and (**GB2+BND**) over that of **VOL**. For *BubbleA'* and *BubbleB'*, the running time of **GB2** is about 9 times and: 2.2-2.3 times that of **VOL**, respectively. For *BubbleC'* the two running times are almost equal (**GB2** being slightly slower).

The value of the ratio $r = n_{\partial O}/n_O$, i.e., the size of the boundary surface of an object O over its inner volume, is the key point to determine whether the boundary-based approach or the volume-based one will be more efficient.

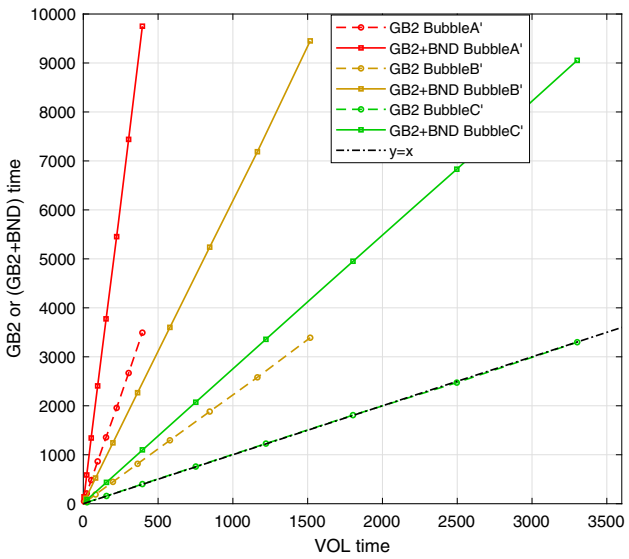


Fig. 16 Running times for the three hollow spheres *BubbleA',B',C'* with the second discretization scheme. The ratios of running times of **GB2** and of **GB2+BND** as a function of the running time of **VOL**

If the surface of the object is already available, we can compute r and, based on its value, select the appropriate algorithm for computing the Euler characteristic.

As the value of the ratio r is inversely dependent on the grid resolution N, this property suggests that the availability

of 3D images with higher and higher resolution will make the surface-based approach more convenient in the future.

6 Concluding Remarks

We explored the computation of the Euler characteristic of a 3D object with the surface-based approach, i.e., by processing only the boundary cells of the cell 3-complex associated with the object, rather than all cells. This requires that the input object is well-composed, and therefore, we considered objects in the BCC grid. We described the classic method based on cell counting and three new ones: one based on an alternative definition of the Euler characteristic provided by Morse theory, and two based on the discrete Gauss–Bonnet theorem.

Among the proposed methods, the one based on discrete Gauss–Bonnet theorem and cycling on faces, outperforms the classic 2D cell counting algorithm by a factor 0.4 for the tested data sets. Experiments have also shown that the boundary-based approach is faster than the classic volumetric one when the number of boundary cells is less than 66% the number of interior cells (or less than 25%, if the boundary has to be extracted just for this purpose), which is likely to happen if the represented 3D shape has no thin parts and the discretization has a sufficient resolution.

The algorithms proposed in this paper are not limited to objects in the BCC grid. The same approaches can be applied to well-composed objects in grids of any type (including well-composed cubic ones), and in general to 3-manifold polyhedral objects. For example, the Gauss–Bonnet theorem based on faces for a well-composed cubic object would compute $\chi(\partial O)$ as $c_0 - c_2$ (as already noted in [8]). We expect that similar results will be obtained experimentally, perhaps with different ratios for discriminating the case when a surface-based computation of the Euler characteristic performs better than a volume-based one.

Author Contributions Lidija Čomić developed the theory behind the algorithms. Paola Magillo developed the technical details of the algorithms and made the experiments. Both authors wrote and reviewed the manuscript.

Funding Open access funding provided by Università degli Studi di Genova within the CRUI-CARE Agreement. Lidija Čomić has received financial support from the Ministry of Science, Technological Development and Innovation of the Republic of Serbia, project number 451-03-47/2023- 01/200156

Data Availability Statement The experiments use input data obtained by discretizing shapes downloaded from the Digital Shape Workbench at http://visionair.ge.imati.cnr.it/ontologies/shapes/

Declarations

Conflict of interest The authors declare that they have no competing interests as defined by Springer, or other interests that might be perceived to influence the results and/or discussion reported in this paper.

Ethical Approval Not applicable. The research does not involve any human and/ or animal studies.

References

1. Banchoff, T.: Critical points and curvature for embedded polyhedral surfaces. Am. Math. Mon. **77**(5), 475–485 (1970). https://doi.org/10.2307/2317380
2. Boutry, N., Géraud, T., Najman, L.: A tutorial on well-composedness. J. Math. Imaging Vis. **60**(3), 443–478 (2018). https://doi.org/10.1007/s10851-017-0769-6
3. Brimkov, V.E., Barneva, R.P.: Analytical honeycomb geometry for raster and volume graphics. Comput. J. **48**, 180–199 (2005). https://doi.org/10.1093/comjnl/bxh075
4. Carvalho, B.M., Garduño, E., Herman, G.T.: Multiseeded fuzzy segmentation on the face centered cubic grid. In: Advances in Pattern Recognition—ICAPR 2001, pp. 339–348 (2001). https://doi.org/10.1007/3-540-44732-6_35
5. Chen, L., Rong, Y.: Linear time recognition algorithms for topological invariants in 3D. In: 19th International Conference on Pattern Recognition (ICPR), pp. 1–4 (2008). https://doi.org/10.1109/ICPR.2008.4761192
6. Chen, L., Rong, Y.: Digital topological method for computing genus and the Betti numbers. Topol. Appl. **157**, 1931–1936 (2010). https://doi.org/10.1016/j.topol.2010.04.006
7. Čomić, L., Magillo, P.: Repairing 3D binary images using the BCC grid with a 4-valued combinatorial coordinate system. Inf. Sci. **499**, 47–61 (2019). https://doi.org/10.1016/j.ins.2018.02.049
8. Čomić, L., Magillo, P.: Surface-based computation of the Euler characteristic in the cubical grid. Graph. Model. **112**, 101093 (2020). https://doi.org/10.1016/j.gmod.2020.101093
9. Čomić, L., Zrour, R., Largeteau-Skapin, G., Biswas, R., Andres, E.: Body centered cubic grid - coordinate system and discrete analytical plane definition. In: Discrete Geometry and Mathematical Morphology DGMM, LNCS, vol. 12708, pp. 152–163 (2021). https://doi.org/10.1007/978-3-030-76657-3_10
10. Csébfalvi, B.: An evaluation of prefiltered B-spline reconstruction for quasi- interpolation on the body-centered cubic lattice. IEEE Trans. Vis. Comput. Graph. **16**(3), 499–512 (2010). https://doi.org/10.1109/TVCG.2009.87
11. Edelsbrunner, H., Harer, J., Zomorodian, A.: Hierarchical Morse–Smale complexes for piecewise linear 2-manifolds. Discrete Com-

put. Geom. **30**(1), 87–107 (2003). https://doi.org/10.1007/s00454-003-2926-5
12. Edelsbrunner, H., Kerber, M.: Dual complexes of cubical subdivisions of $\mathbb{R}^n$. Discrete Comput. Geom. **47**(2), 393–414 (2012). https://doi.org/10.1007/s00454-011-9382-4
13. Finkbeiner, B., Entezari, A., Ville, D.V.D., Möller, T.: Efficient volume rendering on the body centered cubic lattice using box splines. Comput. Graph. **34**(4), 409–423 (2010). https://doi.org/10.1016/j.cag.2010.02.002
14. Hatcher, A.: Algebraic Topology. Cambridge University Press (2001)
15. He, L., Liu, Y., Wang, D., Yun, J.: A voxelization algorithm for 3D body-centered cubic line based on adjunct parallelepiped space. In: Information Computing and Applications—Third International Conference, ICICA, Part I, pp. 352–359 (2012). https://doi.org/10.1007/978-3-642-34038-3_48
16. Ibáñez, L., Hamitouche, C., Roux, C.: Ray-tracing and 3D objects representation in the BCC and FCC grids. In: Discrete Geometry for Computer Imagery, 7th International Workshop, (DGCI), pp. 235–242 (1997). https://doi.org/10.1007/BFb0024844
17. Ibáñez, L., Hamitouche, C., Roux, C.: Ray casting in the BCC grid applied to 3D medical image visualization. In: Proceedings of the 20th Annual International Conference of the IEEE Engineering in Medicine and Biology Society, Vol. 20. Biomedical Engineering Towards the Year 2000 and Beyond (Cat. No.98CH36286), vol. 2, pp. 548–551 (1998). https://doi.org/10.1109/IEMBS.1998.745453
18. Imiya, A., Eckhardt, U.: The Euler characteristics of discrete objects and discrete quasi-objects. Comput. Vis. Image Underst. **75**(3), 307–318 (1999). https://doi.org/10.1006/cviu.1999.0791
19. Kittel, C.: Introduction to Solid State Physics. Wiley, New York (2004)
20. Klette, R., Rosenfeld, A.: Digital Geometry. Geometric Methods for Digital Picture Analysis. Morgan Kaufmann Publishers, San Francisco (2004). https://doi.org/10.1016/S0020-0255(02)00284-0
21. Kong, T., Roscoe, A., Rosenfeld, A.: Concepts of digital topology. Topol. Appl. **46**(3), 219–262 (1992). https://doi.org/10.1016/0166-8641(92)90016-S
22. Kong, T.Y., Roscoe, A.W.: A theory of binary digital pictures. Comput. Vis. Graph. Image Process. **32**(2), 221–243 (1985). https://doi.org/10.1016/S0734-189X(85)80070-0
23. Kong, T.Y., Rosenfeld, A.: Digital topology: introduction and survey. Comput. Vis. Graph. Image Process. **48**(3), 357–393 (1989). https://doi.org/10.1016/0734-189X(89)90147-3
24. Kovalevsky, V.A.: Geometry of Locally Finite Spaces (Computer Agreeable Topology and Algorithms for Computer Imagery). Editing House Dr. Bärbel Kovalevski, Berlin (2008)
25. Labelle, F., Shewchuk, J.R.: Isosurface stuffing: fast tetrahedral meshes with good dihedral angles. ACM Trans. Graph. **26**(3), 57 (2007). https://doi.org/10.1145/1276377.1276448
26. Latecki, L.J.: 3D well-composed pictures. CVGIP Graph. Model Image Process. **59**(3), 164–172 (1997). https://doi.org/10.1016/0167-8655(94)00104-B
27. Lee, C., Poston, T., Rosenfeld, A.: Winding and Euler numbers for 2D and 3D digital images. CVGIP Graph. Model Image Process. **53**(6), 522–537 (1991). https://doi.org/10.1016/1049-9652(91)90003-3
28. Lee, C., Poston, T., Rosenfeld, A.: Holes and genus of 2D and 3D digital images. CVGIP Graph. Model Image Process. **55**(1), 20–47 (1993). https://doi.org/10.1006/cgip.1993.1002
29. Linnér, E.S., Morén, M., Smed, K.O., Nysjö, J., Strand, R.: LatticeLibrary and BccFccRaycaster: software for processing and viewing 3D data on optimal sampling lattices. SoftwareX **5**, 16–24 (2016). https://doi.org/10.1016/j.softx.2016.01.002
30. Matsumoto, Y.: An Introduction to Morse Theory, Translations of Mathematical Monographs, vol. 208. American Mathematical Society (2002)
31. Maunder, C.R.F.: Algebraic Topology. Cambridge University Press (1980)
32. Meng, T., Smith, B., Entezari, A., Kirkpatrick, A.E., Weiskopf, D., Kalantari, L., Möller, T.: On visual quality of optimal 3D sampling and reconstruction. In: Graphics Interface 2007, pp. 265–272. ACM, New York (2007). https://doi.org/10.1145/1268517.1268560
33. Milnor, J.: Morse Theory. Princeton University Press, New Jersey (1963)
34. Ohser, J., Nagel, W., Schladitz, K.: The Euler number of discretized sets-on the choice of adjacency in homogeneous lattices. In: Mecke, K., Stoyan, D. (eds.) Morphology of Condensed Matter. Lecture Notes in Physics, vol. 600. Springer, Berlin (2002). https://doi.org/10.1007/3-540-45782-8_12
35. O'Neill, B.: Elementary Differential Geometry. Elsevier (1966)
36. Strand, R., Nagy, B., Borgefors, G.: Digital distance functions on three-dimensional grids. Theor. Comput. Sci. **412**, 1350–1363 (2011). https://doi.org/10.1016/j.tcs.2010.10.027
37. Toriwaki, J., Yonekura, T.: Euler number and connectivity indexes of a three dimensional digital picture. Forma **17**, 183–209 (2002)

Lidija Čomić is with the Faculty of Technical Sciences, University of Novi Sad, Serbia. Her research interests include digital and computational geometry and topology, image analysis and processing and geometric modeling.

Paola Magillo is an associate professor at the Department of Informatics, Bioengineering, Robotics and Systems Engineering (DIBRIS) of the University of Genova, Italy. Her research interests include computational geometry, geometric modeling, computer graphics, image analysis and processing.