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

Lidija Čomić[a], Paola Magillo[b]

[a]*Faculty of Technical Sciences, University of Novi Sad, Novi Sad, Serbia*
[b]*Department of Computer Science, Bioengineering, Robotics, and Systems Engineering, University of Genova, Genova, Italy*

Abstract

For well-composed (manifold) objects in the 3D cubical grid, the Euler characteristic is equal to half of the Euler characteristic of the object boundary, which in turn is equal to the number of boundary vertices minus the number of boundary faces. We extend this formula to arbitrary objects, not necessarily well-composed, by adjusting the count of boundary cells both for vertex- and for face-adjacency. We prove the correctness of our approach by constructing two well-composed polyhedral complexes homotopy equivalent to the given object with the two adjacencies. The proposed formulas for the computation of the Euler characteristic are simple, easy to implement and efficient. Experiments show that our formulas are faster to evaluate than the volume-based ones on realistic inputs, and are faster than the classical surface-based formulas.

Keywords: Digital topology, Cubical grid, Euler characteristic, Discrete Gauss-Bonnet theorem

1. Introduction

The Euler characteristic is a global topological descriptor of a 3D object O. Intuitively, it is equal to the number of connected components minus the number of tunnels plus the number of cavities [18]. When O is a digital object, made up of voxels in the cubical grid, an appropriate adjacency relation must be considered for voxels in O (i.e., vertex-, edge- or face-adjacency)

Email addresses: comic@uns.ac.rs (Lidija Čomić), magillo@dibris.unige.it (Paola Magillo)

in order to define its Euler characteristic and Betti numbers. The Euler characteristic of O (with vertex-adjacency), and of its boundary ∂O, can be computed locally by counting the number of cells of all dimensions in O and in ∂O, respectively, and taking their alternating sum.

A 3D digital object O is well-composed if the set $|O|$, the union of all voxels in O as a point set, is a 3-manifold (with boundary); in this case, its boundary $|\partial O|$ is a 2-manifold without boundary. The value of the Euler characteristic of a well-composed object does not depend on the chosen adjacency (vertex- or face-adjacency); for such objects it is equal to half the Euler characteristic of the object boundary [8, 15, 22, 26], which in turn, from the discrete Gauss-Bonnet theorem [1], can be computed as $c_0^* - c_2^*$, where c_0^* is the number of vertices and c_2^* is the number of faces in ∂O.

We extend this flowchart (i.e., $\chi(O) = (c_0^* - c_2^*)/2$) from well-composed to arbitrary non-well-composed 3D objects, with respect to both vertex- and face-adjacency, by counting those boundary vertices in ∂O, at which the object is not well-composed, an appropriate number of times. We prove the correctness of our formulas. As a byproduct, we obtain a geometric construction that builds two well-composed polyhedral complexes homotopy equivalent to O with the two (vertex- and face-) adjacencies. Experiments show that our formulas, which count only vertices and faces in ∂O, can be evaluated faster than both the volume-based ones, and the other surface-based formulas, which count all cells in O and in ∂O, respectively.

2. Preliminaries

We give some basic notions on 3D digital topology and well-composed objects, the Euler characteristic, the discrete Gauss-Bonnet theorem and its use in the computation of the Euler characteristic.

2.1. The Cubical Grid

The 3D cubical grid is a tessellation of $\mathbb{R}^3$ into closed unit cubes (called voxels) centered at points in $\mathbb{Z}^3$, with faces parallel to the coordinate planes [17]. Two voxels are face-, edge- or vertex-adjacent (a.k.a. 6-, 18- or 26-adjacent) if they share at least one face, edge or vertex, respectively. They are strictly vertex-adjacent (edge-adjacent) if they share exactly one vertex (edge).

An object O is a finite set of voxels in the cubical grid. The voxels in O are called black (object) voxels; those in the complement of O are called

white (background). Although mathematically the cubical grid is infinite, a finite subset Σ of it is considered, namely a parallelepiped containing O plus at least one layer of additional white voxels in each of the six cardinal directions (i.e., such that the object O does not touch the border of Σ). The complement O^c of O is considered within Σ, and therefore both O^c and O are finite sets.

Two voxels in O are connected (with respect to the chosen adjacency α) if there is a path, composed of voxels in O, starting at one of them and ending at the other, such that any two consecutive voxels in the path are adjacent with respect to α. Connected components of O are the maximal connected subsets of O (with respect to the chosen adjacency). We are interested in objects with vertex- and face-adjacency. If the object is considered with vertex-adjacency, its complement is considered with face-adjacency, and vice versa.

The cubical complex naturally associated with O with respect to vertex-adjacency is composed of all voxels in O, and all their faces, edges and vertices (called cells). With abuse of notation, in the rest of the paper we will use the same letter O to denote the associated complex as well. So, depending on the context, O will denote the set of voxels, or the set of cells of dimensions 0,1,2,3.

The dual complex O^d of O is a cubical complex which has a vertex for each voxel in O, an edge for each pair of face-adjacent voxels in O, a face for each four voxels in O sharing a common edge and a voxel for each eight voxels in O sharing a common vertex.

The underlying space $|O|$ of O is the set of all points contained in some cell in O. The boundary ∂O of O is the complex composed of the faces in O incident to exactly one black voxel, and all their edges and vertices. The topological boundary of O is the underlying space $|\partial O|$ of ∂O.

A vertex v is called critical if it is incident to exactly two black voxels, or exactly two white voxels, and such two voxels intersect only at v (i.e., they have no common edge or face). An edge e is called critical if it is incident to exactly two black voxels, or exactly two white voxels, and such two voxels intersect only at e and at its two endpoint (i.e., they have no common face).

A 3D object O is well-composed [20] if

- $|O|$ is a 3-manifold with 2-manifold boundary $|\partial O|$, or, equivalently [5],
- O contains no critical vertex or edge (i.e., none of the critical configurations illustrated in Figure 1).

Well-composedness is a desirable property for 3D objects. Discrete geometry algorithms, like computation of the distance transform, thinning, or surface tracing, as well as multigrid convergence of some geometric estimators, are faster when applied on well-composed objects. As for applications in geometric modeling and computer graphics, well-composedness allows for implementation of surface parametrization and texture mapping on the object boundary. P: Unfortunately, most real objects, coming from scanning or from discretization of CAD models, are not well-composed.

2.2. The Euler Characteristic

The Euler characteristic is a basic integer-valued topological (homotopical) invariant, extensively used in many application domains. For a 3D well-composed object O, the value of $\chi(O)$ is the same with both the vertex- and the face-adjacency models. For an object which is not well-composed, the value of its Euler characteristic $\chi(O)$ depends on the chosen adjacency relation. We will use the symbols χ_v and χ_f to denote the Euler characteristic according to vertex- and face-adjacency, respectively. The symbol χ will be used when there is no need to distinguish between the two cases. The Euler characteristic can be defined in several equivalent ways. Here, we consider only two:

- through the Betti numbers of O [18] (intuitively, the number β_0 of components, β_1 of tunnels and β_2 of cavities) as

$$\chi(O) = \beta_0 - \beta_1 + \beta_2,$$

- through the number of cells in some complexes associated with O
  - (for vertex-adjacency) through the number c_0 of vertices, c_1 of edges, c_2 of faces and c_3 of voxels in O as

$$\chi_v(O) = c_0 - c_1 + c_2 - c_3. \tag{1}$$

  - (for face-adjacency) [28, 30] through the number b_0 of vertices, b_1 of edges, b_2 of faces and b_3 of voxels in the dual complex O^d as

$$\chi_f(O) = b_0 - b_1 + b_2 - b_3. \tag{2}$$

    Equivalently [28],

$$\chi_f(O) = \chi_v(O^c) - 1. \tag{3}$$

Given an object O (well-composed or not), the Euler characteristic $\chi(\partial O)$ of the boundary ∂O (with vertex-adjacency between the faces) can be computed through the number c_0^* of boundary vertices, c_1^* of boundary edges and c_2^* of boundary faces as

$$\chi(\partial O) = c_0^* - c_1^* + c_2^*. \tag{4}$$

For well-composed objects, there is a connection between the Euler characteristic of O and the Euler characteristic of its boundary ∂O [26] (Proposition 5.4.15) namely

$$\chi(O) = \chi(\partial O)/2. \tag{5}$$

2.3. The Discrete Gauss-Bonnet Theorem

The (continuous) Gauss-Bonnet theorem gives a connection between the Gaussian curvature on a manifold 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, K_G is its Gaussian curvature and the integral is a surface integral over M [29].

The famous 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.,

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

where f are the faces of P 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 theorem of Descartes to (manifold) polyhedra. It states that

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

where P is the given polyhedron and v are the vertices of P [19]. 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.

2.4. The Discrete Gauss-Bonnet Theorem in the Computation of the Euler Characteristic of Well-Composed Objects

The Gauss-Bonnet theorem leads to an efficient way to compute the Euler characteristic of well-composed objects in the cubical grid. For such objects, from (4) and (5), we have

$$\chi(O) = \chi(\partial O)/2 = (c_0^* - c_1^* + c_2^*)/2.$$

As the boundary of a well-composed object O is a polyhedron, $\chi(\partial O)$ can be computed through the discrete Gauss-Bonnet theorem as

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

exploiting the fact that faces are squares and the sum of their internal angles is 2π. Thus

$$\chi(\partial O) = c_0^* - c_2^*. \tag{6}$$

The same formula was obtained by Françon [12] in the context of discrete combinatorial (manifold) surfaces, from (4) and $4c_2^* = 2c_1^*$ (each boundary face is incident to four edges, and each (manifold) boundary edge is incident to two boundary faces). We will apply this formula to arbitrary objects, manifold or not, (with an appropriate modification for counting the boundary vertices depending on the chosen adjacency relation).

3. Related Work

Many algorithms have been proposed for the computation of the Euler characteristic in the 3D cubical grid. The algorithms can be classified according to various criteria:

- the type of adjacency (vertex [2, 18, 19, 24, 28, 31, 32, 39], face [3, 7, 9, 12, 24, 25, 28, 30, 33, 35, 38], or all three types of adjacency [23, 37]),

- the shape of the image (well-composed [7, 9, 12, 27, 33, 35] or not [3, 24, 25, 38]),
- the set of cells processed by the algorithm (only border cells [16, 21, 24, 25], or all cells, including the interior ones [2, 3, 7, 18, 19, 30, 31, 32, 33, 35, 37, 38, 39]),
- the processed complex (the given one [2, 3, 9, 16, 18, 19, 21, 25, 31, 32, 37, 39], or its dual [7, 28, 30, 33, 35, 37, 38]),
- the adopted definition of the Euler characteristic (the alternating sum of the number of i-cells [12, 18, 19, 24, 25, 30, 37, 38, 39], the alternating sum of the Betti numbers [31, 32]),
- the use of a compressed data structure for storing the image (run-length encoding [23], or bintree [2]),
- the theoretical basis of the algorithm (winding number and Morse theory [21] or discrete Gauss-Bonnet theorem, either directly [9] or by introducing the notion of the curvature index of the vertices in the boundary [16]).

We call volume-based approaches those considering all cells in O; we call surface-based the ones involving only the cells in ∂O. We restrict our attention to methods having some relation with ours, i.e., those which use a surface-based approach on arbitrary objects (not necessarily well-composed) and those based on the discrete Gauss-Bonnet theorem.

The algorithms [24, 25] that compute the Euler characteristic of an arbitrary object O by counting the cells in ∂O (equation (4)) adjust the vertex and edge count for non-manifold configurations to get a correct result for face-adjacency. P: [not here, in section 6] [DELETE These algorithms are slower than ours, because they count edges as well.]

The discrete Gauss-Bonnet theorem has been used for well-composed objects. Chen and Rong [9] apply the discrete Gauss-Bonnet theorem to connected manifold objects with no cavities. The number of vertices incident to k boundary faces is denoted m_k, $3 \leq k \leq 6$. 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 $\delta(v) = 2\pi - k\pi/2$. The discrete Gauss-Bonnet theorem implies that

$$\chi(\partial O) = (m_3 - m_5 - 2m_6)/4.$$

Imiya and Eckhardt [16] use the same formula, which they obtain through a finer classification of the boundary vertices.

Lee at al. [21] proposed an algorithm for arbitrary objects, based on smoothing the voxels (slightly inflating them and rounding the corners and edges) and applying the continuous Gauss-Bonnet theorem. The proposed algorithm can handle objects with vertex- and face-adjacency (by considering the complement of the object). It reduces to using a look-up table with vertex contributions to $\chi(O)$ for each possible configuration of $2\times2\times2$ cubes incident to the vertex.

In Section 6, we will compare experimentally our formula with one representative of the volume-based approach and one representative of the surface-based approach. The former will be a modified Equation (1), which is also used in [37, 39]. The latter will be the formula used by Lozano-Durán and Borrell [25].

4. The Proposed Formulas

We compute the Euler characteristic $\chi(O)$ of an object O through the Euler characteristic $\chi(\partial O)$ of its boundary ∂O as $\chi(O) = \chi(\partial O)/2$ with $\chi(\partial O)$ in turn computed as $\chi(\partial O) = c_0^* - c_2^*$ (see Equations (5) and (6), respectively). For well-composed objects, this is the known formula. For arbitrary objects, we adjust the count of the number c_0^* of boundary vertices to get the correct result, maintaining the duality between one adjacency for black voxels and the opposite adjacency for white ones, as described by Equation (3).

For computing both χ_v and χ_f, manifold (non-critical) vertices are counted once. Non-manifold vertices are in one of the eleven critical configurations shown in Figure 1.

Vertex adjacency. Our modified Equation (4) for an object O with vertex-adjacency is

$$\chi_v = c_0^{*v} - c_2^* \tag{7}$$

where the symbol c_0^{*v} denotes the number of boundary vertices, counting each non-manifold boundary vertex as many times as there are face-connected components of white voxels incident to it, with the exception of the critical vertices incident to two strictly vertex-adjacent black voxels (and six white ones), which are not counted. Referring to Figure 1, the central vertex is

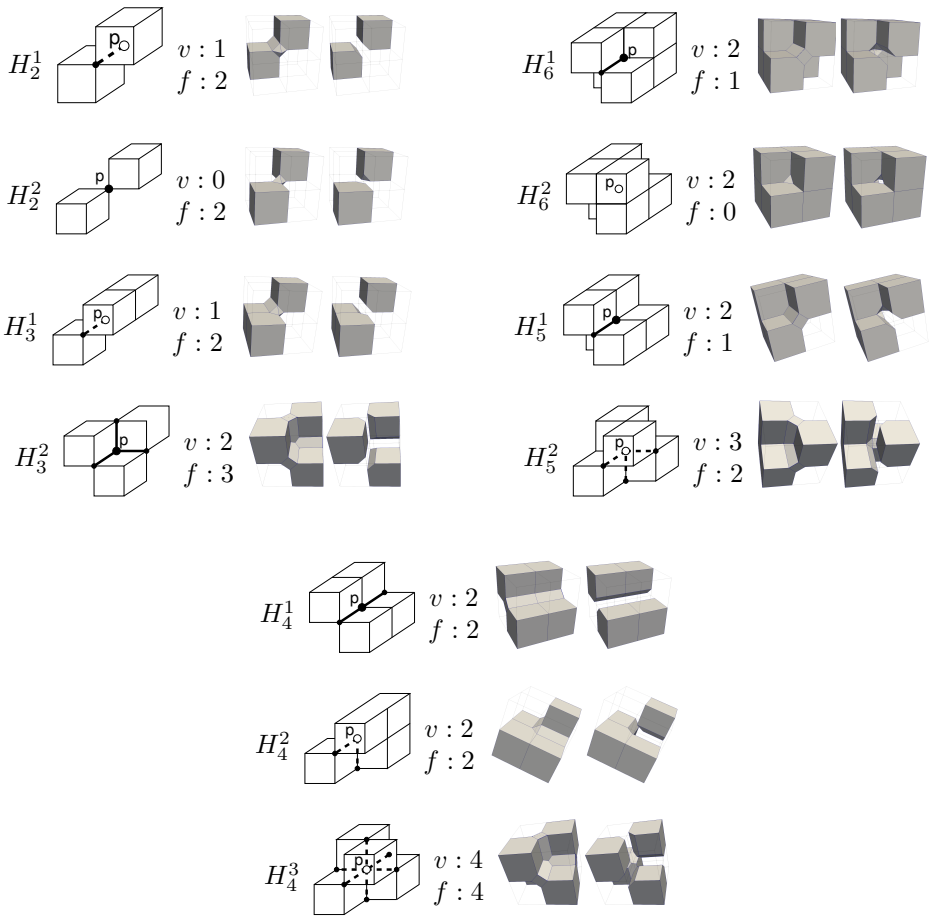


Figure 1: In the first column, the possible configurations (up to rotation and symmetry) of black voxels around a non-manifold boundary vertex. In notation H_k^p, k denotes the number of black voxels, and p is an index discriminating the configurations with the same number of black voxels. The second column reports how many times the central vertex is counted with vertex-adjacency (v:) and face-adjacency (f:). The configurations H_k and H_{8-k}, with the same superscript, are mutually dual and are shown on the same line. The last two columns show polyhedra P_v and P_f used in the proof of correctness. For clarity, the images show only polyhedra associated with cells incident to the central vertex (the octahedra corresponding to other vertices are not shown).

counted once in configurations H_2^1 and H_3^1, three times in H_5^2, four times in H_4^3, zero times in H_2^2 and two times in the remaining configurations.

Face adjacency. Our modified Equation (4) for an object O with face-adjacency is

$$\chi_f = c_0^{*f} - c_2^* \tag{8}$$

where the symbol c_0^{*f} denotes the number of boundary vertices, counting each non-manifold boundary vertex as many times as the number of face-connected components of black voxels incident to it, with the exception of the critical vertices incident to two strictly vertex-adjacent white voxels (and six black ones), which are not counted. In Figure 1, the central vertex is counted once in configurations H_5^1 and H_6^1, three times in H_3^2, four times in H_4^3, zero times in H_6^2 and two times in the remaining configurations.

5. The Proof of Correctness

We give a formal proof that our formulas correctly compute the Euler characteristic with vertex- or face-adjacency. As a byproduct of the proof, we obtain a geometric construction, which transforms an object O in the cubical grid into two well-composed polyhedra P_v and P_f, homotopy equivalent to O with vertex- and face-adjacency, respectively. We describe the construction of the two polyhedra, we prove their well-composedness, we show the homotopy equivalence, and we show that the alternating sum of boundary cells in P_v and P_f (i.e., their Euler characteristic) is equal to the count $c_0^{*v} - c_2^*$ and $c_0^{*f} - c_2^*$, respectively (i.e., counting c_0^* with appropriate multiplicities for non-manifold vertices), as in our formulas (7) and (8).

5.1. Construction of the Polyhedra P_v and P_f

The constructions of the two polyhedra P_v and P_f from the object O with vertex- and face-adjacency, respectively, are dual: P_v is obtained by applying to white voxels the same construction that produces P_f if applied on black ones and vice versa.

For an arbitrary length a, with $0 < a < \sqrt{2}/2$, we associate an octahedron with each vertex v, centered at v, with vertices on the coordinate axes through v, and triangular faces of side a.

We associate a square prism with each edge e, with square bases of side a in the axis-orthogonal planes through the endpoints of e, and with vertices on the coordinate axes through the endpoints.

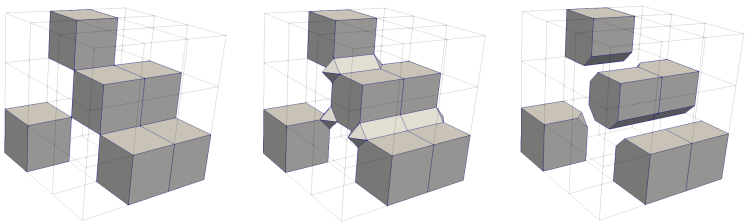


Figure 2: From left to right: an object O and polyhedra P_v and P_f.

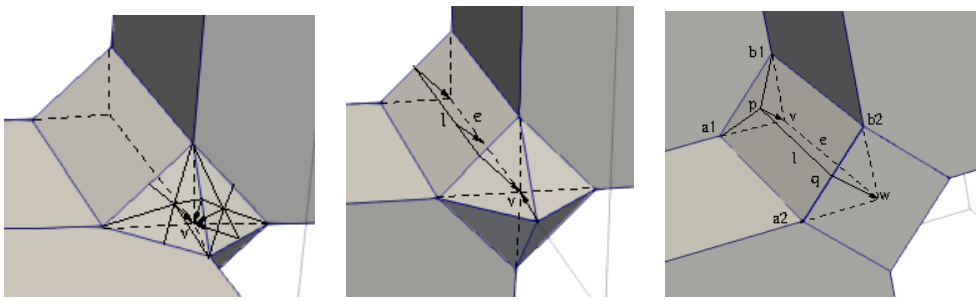
Vertex-Adjacency. We construct the polyhedron P_v by thickening O at all non-manifold edges and vertices.

- For each central vertex v of a critical configuration, we add the octahedron associated with v, i.e., we add the tetrahedra corresponding to the parts of the octahedron inside each white voxel incident to v.
- For each critical edge e, we add the prism associated with e, i.e., we add the triangular prisms corresponding to the parts of that prism lying inside each white voxel incident to e.

Face-Adjacency. We construct the polyhedron P_f by disconnecting O at all non-manifold edges and vertices.

- For each central vertex v of a critical configuration, we remove the octahedron associated with v, i.e., we truncate each black voxel incident to v, by removing the part (a tetrahedron) of the octahedron inside that voxel.
- For each critical edge e, we remove the prism associated with e, i.e., we truncate each black voxel incident to e by removing the part (a triangular prism) of the square prism inside that voxel.

In both cases, we remove and add the tetrahedra and prisms as point sets, and then we subdivide the boundaries of the obtained polyhedra in the obvious manner. This construction is illustrated in Figure 2, and the result for each of the eleven critical configurations is illustrated in Figure 1.



(a) (b) (c)

Figure 3: Construction of the deformation retraction (homotopy) from the polyhedron P_v onto the object O.

5.2. Well-Composedness

Proposition 1. *The (underlying spaces of the) polyhedra P_v and P_f are 3-manifolds, with 2-manifold boundaries (i.e., they are well-composed).*

PROOF. The parts of the polyhedra that coincide with the given object O are manifold, because the construction affects only the neighborhood of non-manifold vertices and edges in O (their 1/2-neighborhood, i.e., the set of points at distance less than 1/2 from such vertices or edges). The manifoldness of the polyhedra in these neighborhoods follows from inspecting the polyhedra (see Figure 1), and from the choice of a ($0 < a < \sqrt{2}/2$), which prevents the vertices and edges of the added / deleted octahedra and prisms from becoming non-manifold.

5.3. Homotopy Equivalence

Proposition 2. *The polyhedron P_v is homotopy equivalent to the object O with vertex-adjacency.*

PROOF. We construct a map that pushes in the protruding parts of P_v (tetrahedra and prisms) onto O.

- We barycentrically subdivide each equilateral triangle in ∂P_v, and we map the center of each triangle and the center of each of its protruding edges on the center v of the corresponding octahedron (see Figure 3(a)). In H_2^1 we map only the protruding vertex of the octahedron to v (see Figure 3(b)).

- We introduce the midline l (parallel to the corresponding critical edge e) into faces in ∂P_v originating from prisms, and we map l linearly onto e (see Figure 3(b)). For H_5^1 and H_6^1, to avoid discontinuities, we introduce first a point p on l at distance e.g. a from the center v of the critical configuration, we map p to v and we map linearly the part of l away from v onto e (see Figure 3(c)).

- We extend these maps linearly to the rest of the protruding parts of P_v.

This map is well-defined: each vertex in P_v has a unique image (since $a < \sqrt{2}/2$). It is continuous, and induces a homotopy (a deformation retraction) between P_v and O.

The homotopy equivalence of P_f to O with face-adjacency follows by duality.

The two polyhedra P_v and P_f can be seen as two repaired versions of the given object O, with vertex- and face-adjacency, respectively. Object repairing is the process of transforming a given object O into another one, which is well-composed and similar, in some way, to O. Homotopy equivalence can be one of the requirements for such similarity. There is a wide literature on image repairing in the cubical grid. Some repairing methods produce a well-composed object which is still cubical [34, 36], others produce a more general polyhedral object [10, 11, 13, 14]. It is worthwhile to analyse the possible advantages of a use of polyhedra P_v and P_f in repairing (this would give a method belonging to the second class), but it is out of the scope of the present paper.

5.4. Correctness of the formulas

Proposition 3. *Formula (7) correctly computes the Euler characteristic of a given object O with vertex-adjacency and of its boundary ∂O.*

PROOF. We compute the contribution of each (central) vertex v to the sum $c_0^* - c_2^*$ in ∂O, as in formula (7). We also compute the contribution of the corresponding part of ∂P_v to the sum $d_0^* - d_1^* + d_2^*$ in ∂P_v (the star denotes boundary cells; c are cells in ∂O, d are cells in ∂P_v). We count each (manifold) vertex in ∂P_v once, for each vertex we count each incident edge $1/2$ times, and for each vertex we count each incident k-gon $1/k$ times; the rest of the edges and faces are counted with the remaining incident vertices. The

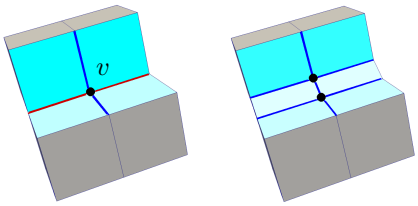


Figure 4: Illustration of the proof of correctness for configuration H_4^1: the object O (left) and the polyhedron P_v (right). In P_v, v has been replaced by four vertices, only two of them are visible in the image.

two contributions, shown in Table 1, are equal for each of the possible configurations in Figure 1, implying that the quantity $c_0^* - c_2^*$ computed by Formula (7) is equal to $\chi(\partial P_v)$. The correctness of the formula follows from homotopy equivalence.

We give details for the case H_4^1 (see Figure 4); the remaining cases can be derived in a similar manner. We denote by Δc_i and Δd_i the contributions of the critical vertex to the numbers c_i and d_i, respectively, which are summed up to compute the Euler characteristic. In H_4^1, the central vertex v is counted two times in ∂O ($\Delta c_0^* = 2$), because v is non-manifold and incident to two face-connected components of white voxels. The vertex v is incident to eight squares, and we count 1/4 of each square ($\Delta c_2^* = 2$). Thus, $\Delta(c_0^* - c_2^*) = 0$.

In ∂P_v, the central vertex v is transformed to four vertices ($\Delta d_0^* = 4$). Each of them is incident to four edges in ∂P_v (two edges parallel to a critical edge incident to v, one edge of length a, and one originating in a non-critical edge in ∂O). We count each of these edges 1/2 times ($\Delta d_1^* = 8$). Each of the four vertices is incident to four rectangular faces in ∂P_v (two originating from the prisms, and the other two from the squares in ∂O). Each of the faces is counted 1/4 times ($\Delta d_2^* = 4$). Thus, $\Delta(d_0^* - d_1^* + d_2^*) = 0 = \Delta(c_0^* - c_2^*)$.

The correctness of Formula (8) follows by duality.

6. Experiments

We compare our proposed formulas with the ones at the state of the art.

6.1. Implementation

We have implemented and tested the following formulas for computing the Euler characteristic of an object O in the cubical grid:

Table 1: Local cell count for the computation of $\chi(\partial O)$ with our rules for the configurations in Figure 1, and for $\chi(\partial P_v)$ with the usual formula valid for 2-manifolds for the corresponding polyhedra.

	c_0^*	c_2^*	$c_0^* - c_2^*$	d_0^*	d_1^*	d_2^*	$d_0^* - d_1^* + d_2^*$
H_2^1	1	3/2	−1/2	5	12	13/2	−1/2
H_2^2	0	3/2	−3/2	6	15	15/2	−3/2
H_3^1	1	7/4	−3/4	5	23/2	23/4	−3/4
H_3^2	2	9/4	−1/4	7	27/2	25/4	−1/4
H_4^1	2	2	0	4	8	4	0
H_4^2	2	2	0	5	10	5	0
H_4^3	4	3	1	16	24	9	1
H_5^1	2	7/4	1/4	5	17/2	15/4	1/4
H_5^2	3	9/4	3/4	6	21/2	21/4	3/4
H_6^1	2	3/2	1/2	4	6	5/2	1/2
H_6^2	2	3/2	1/2	6	9	7/2	1/2

VOL Equation (1), i.e., the alternating sum of the number of vertices, edges, faces and voxels, computed on O. As used in [37, 39], this formula applies for vertex- or face-adjacency. For face-adjacency, c_1 counts each non-manifold edge twice; c_0 counts each vertex v as many times as the number of face-connected black components within the eight voxels incident in v, with the exception of a critical vertex v incident to two strictly vertex-adjacent white voxels (H_6^2 in Figure 1), which is not counted. We denote as c_0^f and c_1^f such adjusted values. So, VOL splits into two formulas VOL_v and VOL_f, for vertex- and face-adjacency, respectively. By Equation (1), $\mathrm{VOL}_v = c_0 - c_1 + c_2 - c_3$, while $\mathrm{VOL}_f = c_0^f - c_1^f + c_2 - c_3$.

LDB Equation (4), i.e., the alternating sum of the number of boundary vertices, edges and faces, computed on ∂O. This formula applies for face-adjacency, as used in [25], with adjusted value of c_0^* and c_1^*. Each critical edge is counted twice. Here, we count critical vertices incident to six black and two strictly vertex-adjacent white voxels zero times, while in [25] they seem to be counted twice (but then configuration H_6^2 would get wrongly $\chi(\partial O) = 26 - 48 + 24 = 2$). With this modification, the computation of c_0^* is the same as c_0^{*f} in our approach for face-adjacency. So, LDB can be written as $c_0^{*f} - c_1^{*f} + c_2^*$.

OUR Equation (6) based on discrete Gauss-Bonnet theorem, i.e., the alternating sum of the number of boundary vertices and faces, computed on ∂O. Our proposed Formulas apply (6) with adjusted value of c_0^*, i.e., c_0^{*v} and c_0^{*f} for vertex- and face-adjacency, respectively. Thus, OUR splits into two formulas: OUR_v, i.e., Equation (7) for vertex-adjacency, and OUR_f, i.e., Equation (8) for face-adjacency.

In order to provide a program for Euler computation, formulas have to be inserted into an algorithm. Details of the algorithm, such as the approach used to scan the grid containing the object, and optimizations, such as the use of look-up tables, have a strong impact on the performance.

Exploiting the fact that the black/white status of the $2 \times 2 \times 2$ voxels incident to a vertex can be encoded in an 8-bit mask, many proposed algorithms use a look-up table with $2^8 = 256$ entries, containing the contribution to χ of each vertex configuration.

This optimization is used in [37] and in [25] (the latter using three look-up tables, which could be reduced to one), and it can be used in our approach as well, in the following way. For each configuration of $2 \times 2 \times 2$ cubes, we can store the difference of the vertex count (1 for manifold boundary vertices, the number in Figure 1 for non-manifold boundary vertices, and 0 for all other vertices), minus a face count equal to $\frac{1}{4}$ the number of boundary faces incident to the vertex (as the same face is counted with its four vertices). In order to keep integer values, we can store this number multiplied by four, the algorithm will compute 4χ and divide it by four at the end.

By using a look-up table, all formulas will compute χ as the sum of the values found in the table, for each vertex existing in the input image, and therefore the mathematical part of the computation will take the same time. The actual running time of each algorithm will then be determined by the strategy used to scan vertex configurations. The simplest way is a sequential scan of all $2 \times 2 \times 2$ configurations in the grid Σ. Note that some configurations have an entry equal to zero (i.e., they do not contribute to the Euler characteristic). Volume-based methods will benefit from scanning techniques visiting configurations with black cubes only (possibly organized in runs or trees). Surface-based methods will benefit from a knowledge of the object boundary (e.g., in the form of a contour code).

Considering the best possible solution, where only configurations contributing to χ are visited, the computational time complexity of the volume-based approach (VOL) is linear in the number c_3 of black voxels, as it iterates

Table 2: Running times (in milliseconds) of VOL, LDB, and OUR on *Solid block* (the best case, a block of N^3 black voxels), and *Chessboard* (the worst case, a three-dimensional chessboard of side N). Both inputs are for $N = 64, 128, 256, 512$.

	N	c_3	VOL_v	VOL_f	LDB	OUR_v	OUR_f
Solid	64	262,144	406	486	118	79	98
box	128	2,097,152	3,233	3,868	623	472	471
	256	16,777,216	25,907	30,953	3,825	3,212	3,200
	512	134,217,728	216,645	257,210	33,754	31,241	31,249
Chess-	64	131,072	228	321	473	246	246
board	128	1,048,576	1,828	2,555	3,760	1,951	1,953
	256	8,388,608	14,673	20,444	30,150	15,637	15,670
	512	67,108,864	126,878	171,086	249,301	133,646	134,037

on all black voxels. That of the surface-based approach (LDB and OUR), instead, is linear in the number of black cubes having some face (from 1 to 6 faces) on the boundary, which has the same order of magnitude as the number c_2^* of square faces shared by a black and a white voxel (boundary faces).

Here, we want to compare just the used formula, not the algorithmic setting surrounding it. Therefore, we adopt a simple scan of all black cubes. From them, we access faces, edges, and vertices, as required by the specific formula (VOL, LDB or OUR). We use marks to avoid counting twice the same face, edge or vertex. We use a look-up table only to store the number of times a vertex should be counted, equally in VOL, LDB, and OUR.

6.2. Experimental results

Algorithms have been implemented in C language and executed on a PC equipped with an Intel CPU i7-2600K CPU at 3.4 Gigahertz with 32 Gigabytes of RAM. They have been run ten times on each input, taking the smallest execution time.

Before describing the experiments, we introduce the ratio $r = c_2^*/c_3$, that will be used in the analysis of the results. We expect that r, expressing the size of the boundary surface over the size of the volumetric object, will impact on the relative performance of surface-based methods (LDB and OUR) and volume-based ones (VOL). In order to analyze the performance in limit cases, we first considered two synthetic objects, corresponding to the maximum and minimum ratio r. *Solid box* is a block of N^3 black voxels and is the best case, having the largest volume and the smallest boundary surface. The number of boundary faces is $c_2^* = 6N^2$, and $r = 6/N$ decreases while increasing N. The

object is well-composed and homeomorphic to a ball, the Euler characteristic is $\chi = 1$. *Chessboard* is a three-dimensional chessboard of edge N. This is the worst case, having the smallest volume and the largest boundary surface. There are $N^3/2$ black voxels (for even N, as it is in our experiments), and all their faces lie on the boundary, $r = 6$. The object is not well-composed. With vertex-adjacency, there is a single connected component and $(N - 2)^3/2$ cavities, thus the Euler characteristic is $1 + (N - 2)^3/2$. With face-adjacency, each black voxel is a connected component on its own, and the Euler characteristic is $N^3/2$.

These two synthetic objects have been generated with $N = 64$, 128, 256, 512, and Table 2 shows the running times for computing the Euler characteristic with each of them. The performance of OUR on real inputs will be between such two limit cases.

The version of VOL with vertex-adjacency is always faster than the one with face-adjacency, as it does not require a special processing of non-manifold vertices. For OUR, the two versions have no relevant difference in execution time. As expected, on *Solid box* the surface-based approach to Euler computation (LDB and OUR) clearly outperforms the volume-based one (VOL), because it counts fewer cells. For *solid box*, OUR is 5 to 9 times faster than VOL with the corresponding adjacency type; OUR with face-adjacency is 1.08 to 1.3 times faster than LDB.

Conversely, on *Chessboard* the volume-based approach with vertex-adjacency is the fastest one. Here, every configuration of $2 \times 2 \times 2$ cubes in Σ is on the boundary. Differences in running times depend on the time for evaluating the used formula (i.e., for accessing faces, edges and vertices to be counted). With vertex-adjacency, the volume-based approach is faster than OUR, while with face-adjacency OUR is still faster than VOL. Ratios between execution times with the two approaches are smaller here, ranging from 0.9 to 1.3. LDB is the slowest one, taking a bit less than twice the running time of OUR with face-adjacency. P: This is due to the fact that LDB has to count edges as well.

We executed VOL, LDB and OUR on two realistic test sets. The first one, called *Shapes*, consists of about one hundred objects derived from shapes present in the Digital Shape Workbench[1]. The original shapes were in vector format, either as tetrahedral meshes, or as triangle meshes (defining oriented

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

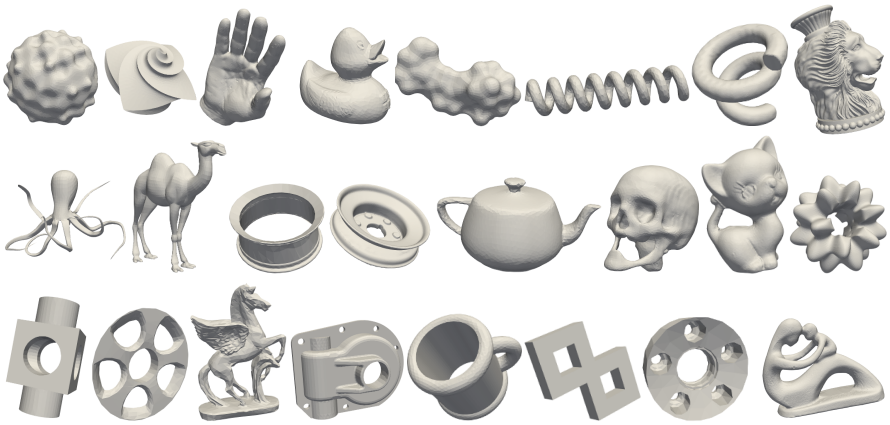


Figure 5: Some of the test shapes that have been discretized on a cubical grid.

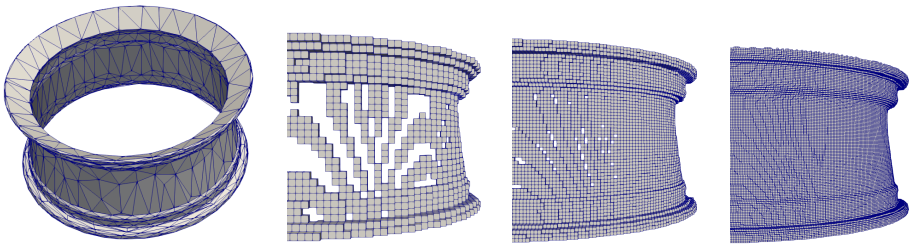


Figure 6: A test shape and details of its discretizations for $N = 64, 128, 256$. The given shape has very thin walls and the lowest resolutions are not sufficient for a correct discrete representation.

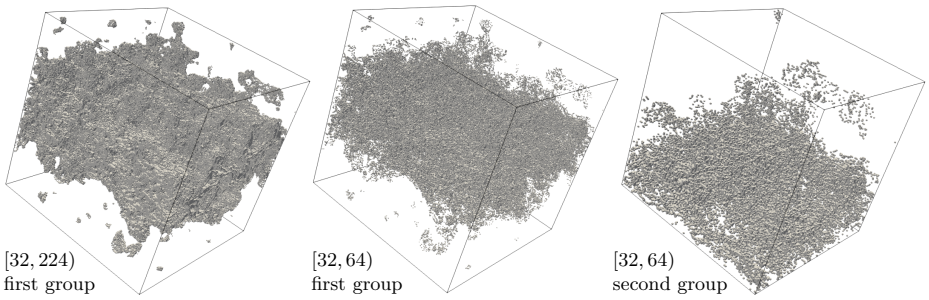


Figure 7: Some of the inputs obtained from *Perlin noise*, gray levels in the indicated intervals have been considered as black.

closed surfaces). Some examples are shown in Figure 5. We superimposed a cubical grid of N^3 voxels and marked a voxel as black if its center falls inside the volume covered by the tetrahedral mesh, or bounded by the triangle mesh. Discretizations have been performed with $N = 64, 128, 256, 512$. For a fixed shape, r halves when we double the grid size N (c_3 is roughly multiplied by 8, c_2^* by 4). The resulting objects are generally not well-composed, and their Euler characteristic may not coincide with that of the original shape, due to the limited resolution (parts may be disconnected, holes closed, connected components merged, etc.). Examples of discretizations are shown in Figure 6. The value of the ratio r is between 0.02 and 2.7.

The second test set, called *Perlin*, is obtained from a gray-scale 3D image representing Perlin noise[2] on a grid of size $N = 256$, with gray levels from 0 to 255. From it, we obtained 20 objects by selecting ranges of values to be black (some examples are shown in Figure 7). Such objects, that we will refer to as first group, are near to the worst case, with ratio r between 2.7 and 5.4 (i.e., greater than in test set *Shapes* and near to the maximum $r = 6$ achieved by *Chessboard*). Another 20 objects, that we will refer to as second group, have been obtained by selecting just 1/8 of each object of the first group, and replacing each black voxel with a block of eight black voxels. This gives a comparable number of black voxels (because of the uniform structure of Perlin noise), and about half the number of faces. Thus the values of r in the second group are about half of those in the first group, and they range from

[2] `https://www.thingiverse.com/thing:27229`

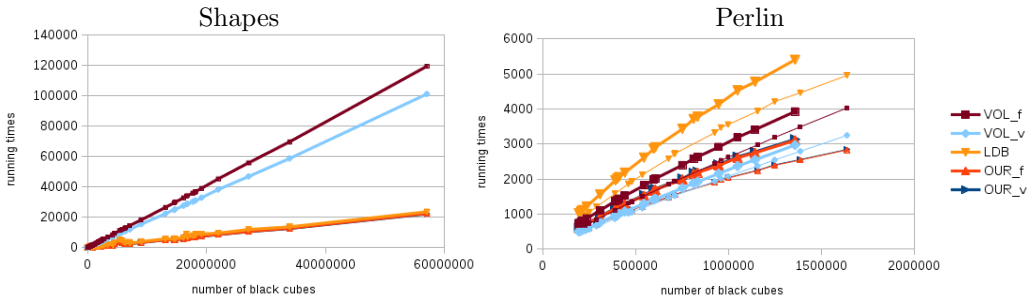


Figure 8: Running times of VOL, LDB, and OUR on the two test sets *Shapes* and *Perlin*. The horizontal axis is the number c_3 of black voxels, the vertical axis is execution time in milliseconds. For *Perlin*, the two curves correspond to the two groups of inputs. See text for comments.

1.3 to 2.7. All objects in test set *Perlin* have many connected components, tunnels and cavities (in the order of 10^2 or 10^3).

On all objects, $\chi(\partial O)$ computed by OUR is exactly twice $\chi(O)$ computed by VOL with the corresponding adjacency, and $\chi(\partial O)$ computed by LDB and by OUR with face-adjacency are equal.

Figure 8 shows the running times on the two test sets, as a function of the number c_3 of black voxels. Figure 9 shows the ratio of execution times of VOL and LDB over the execution time of OUR, with the same adjacency type, illustrating their dependency on r. A larger time ratio means that the implementation using our formulas is faster. In the following we comment such results for the two test sets.

For test set *Shapes*, in Figure 8 the curves for the three surface-based formulas (LDB, OUR_v, OUR_f) are almost superimposed on each other, and the surface-based approach gives much lower execution times than the volume-based one. As expected, we see a linear dependency of the running times of VOL on c_3. The expected sub-linear dependency of LDB and OUR cannot be appreciated here (it will be visible for test set *Perlin*) because too many parameters impact on the value of c_2^* beside c_3, such as the thickness of the object shape and the grid side N.

Still for *Shapes*, in Figure 9 OUR can be up to more than five times faster than VOL for small values of r, but this improvement becomes smaller for larger values of r. For most inputs, having $r < 1$, OUR is over 1.5 times

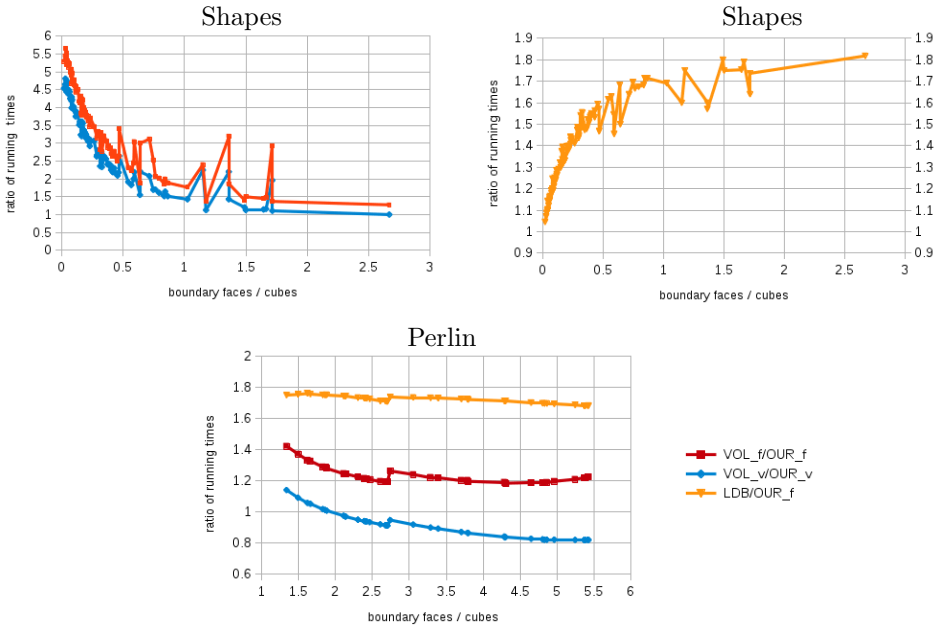


Figure 9: Ratios of the running times of VOL and LDB over the running time of OUR. The vertical axis is the ratio of the running times, the horizontal axis is the ratio $r = c_2^*/c_3$. For test set *Shapes*, VOL and LDB give very different ratios over OUR and are plotted separately. For test set *Perlin*, they are plotted together in the same scale. See text for comments.

faster than VOL. Inputs with $r > 1$ are relatively few and at the lower grid resolutions ($N = 64$ or 128). The input giving the minimum gain in speed ($r = 2.667179$ and time ratio VOL/OUR equal to 1 with vertex-adjacency), is the one shown in Figure 6 (b), but this discretization does not have a sufficient resolution to correctly represent the corresponding shape. All other inputs with $r > 1$ correspond to similar situations. Thus, in some sense, OUR is not faster than VOL in cases where, in practice, the resolution is insufficient for a correct shape representation and, thus, for reliable computation of the Euler characteristic. Figure 9 shows that, for *Shapes*, OUR with face-adjacency is between 1 and 1.9 times faster than LDB, with better performance for larger r, thus for shapes more distant from that of a solid block.

In summary, for test set *Shapes*, our proposed formulas are always faster to evaluate than the other ones. The speed-up with respect to VOL decreases with grid resolution, and the speed-up with respect to LDB increases with grid resolution. From these results, we expect that our method will be especially convenient for shapes with moderate complexity represented at medium resolution (such that the ratio r is between 0.25 and 0.5), which seems a realistic scenario.

For test set *Perlin*, in Figure 8 we show two separate curves for the first test group (with larger r, thick lines) and for the second one (with smaller r, thin lines). The curves for OUR with vertex- and with face-adjacency are almost superimposed on each other, while the curve for LDB is above all the others. All thick lines are above the corresponding thin line, i.e., increasing r increases the running time as well. As the grid side N is fixed, within each group r decreases with increasing c_3. Figure 8 for *Perlin* shows a linear trend of the running time of VOL, and a slightly sub-linear trend for the surface-based LDB and OUR, as expected.

In test set *Perlin*, the convenience of our approach over the others decreases dramatically, as expected. LDB has the largest execution times. With vertex-adjacency, OUR is slower than VOL, with the exception of some inputs of the second group. With face-adjacency, OUR is slightly faster than VOL. In Figure 9, the first group of the *Perlin* set has $r > 2.7$ and the second one $r < 2.7$ (a step is visible in the curves). With vertex-adjacency, on the first group VOL has always the best running time, which is little more than 0.8 of the time of OUR; on the second group, OUR is faster for $r < 2$, but the speed-up is < 1.2. With face-adjacency, OUR is clearly faster than LDB (over 1.6 times), and slightly faster than VOL (about 1.2 times for the first group, and up to 1.4 times for the second one). Time ratios tend to become

stable for r approaching 6: LDB/OUR tends to 1.7, VOL/OUR tends to 0.8 with vertex-adjacency and to 1.2 with face-adjacency.

In summary, OUR is always preferable to LDB. It is evaluated faster than VOL for objects having a value of $r < 2$, and may be up to six times faster for small r. If the object is a discretized 3D shape at sufficient resolution, then OUR is always convenient over other methods. If the object represents a porous medium, and the Euler characteristic is to be computed with face-adjacency, then our method leads to a moderate time saving, while with vertex-adjacency the volumetric approach is faster.

7. Summary

We proposed a surface-based formula for computing the Euler characteristic of an arbitrary object (well-composed or not) in the cubical grid, with either vertex- or face-adjacency. The formula is based on counting only the boundary vertices and faces in the object, with the vertex count adjusted for the two adjacency relations. As objects coming from discretizations are rarely well-composed, this makes our approach very useful. Our new surface-based formula is conceptually simple and easy to implement.

A formal proof of correctness of the proposed formulas transforms the input object O into two well-composed polyhedral complexes, homotopy equivalent to O, depending on the chosen adjacency. Thus, it leads to another possible way to repair an object O in the cubical grid, adding a new member to the rapidly growing family of repairing algorithms [4, 6, 10, 11, 13, 14, 34, 36].

Acknowledgments

This work has been partially supported by the Ministry of Education and Science of the Republic of Serbia within the Project No. 34014.

References

[1] Banchoff, T., 1970. Critical Points and Curvature for Embedded Polyhedral Surfaces. American Mathematical Monthly 77 (5), 475–485.

[2] Bieri, H., 1987. Computing the Euler characteristic and related additive functionals of digital objects from their bintree representation. Computer Vision, Graphics, and Image Processing 40 (1), 115–126.

[3] Bieri, H., Nef, W., 1984. Algorithms for the Euler characteristic and related additive functionals of digital objects. Computer Vision, Graphics, and Image Processing 28 (2), 166–175.

[4] Boutry, N., Géraud, T., Najman, L., 2015. How to make nD images well-composed without interpolation. In: 2015 IEEE International Conference on Image Processing, ICIP 2015. pp. 2149–2153.

[5] Boutry, N., Géraud, T., Najman, L., 2018. A Tutorial on Well-Composedness. Journal of Mathematical Imaging and Vision 60 (3), 443–478.

[6] Boutry, N., González-Díaz, R., Jiménez, M.-J.., 2019. Weakly well-composed cell complexes over nD pictures. Information Sciences 499, 62–83.

[7] Bribiesca, E., 2010. Computation of the Euler number using the contact perimeter. Computers & Mathematics with Applications 60 (5), 1364–1373.

[8] Chen, L., Rong, Y., 2008. Linear time recognition algorithms for topological invariants in 3D. In: 19th International Conference on Pattern Recognition (ICPR. pp. 1–4.

[9] Chen, L., Rong, Y., 2010. Digital topological method for computing genus and the Betti numbers. Topology and its Applications 157, 1931–1936.

[10] Čomić, L., Magillo, P., 2019. Repairing 3D binary images using the BCC grid with a 4-valued combinatorial coordinate system. Information Sciences 499, 47–61.

[11] Čomić, L., Magillo, P., 2019. Repairing 3D Binary Images Using the FCC Grid. Journal of Mathematical Imaging and Vision 61 (9), 1301–1321.

[12] Françon, J., 1995. Discrete Combinatorial Surfaces. CVGIP: Graphical Model and Image Processing 57 (1), 20–26.

[13] González-Díaz, R., Jiménez, M.-J.., Medrano, B., 2015. 3D well-composed polyhedral complexes. Discrete Applied Mathematics 183, 59–77.

[14] González-Díaz, R., Jiménez, M.-J.., Medrano, B., 2017. Efficiently Storing Well-Composed Polyhedral Complexes Computed Over 3D Binary Images. Journal of Mathematical Imaging and Vision 59 (1), 106–122.

[15] Hatcher, A., 2001. Algebraic Topology. Cambridge University Press.

[16] Imiya, A., Eckhardt, U., 1999. The Euler Characteristics of Discrete Objects and Discrete Quasi-Objects. Computer Vision and Image Understanding 75 (3), 307–318.

[17] Klette, R., Rosenfeld, A., 2004. Digital geometry. Geometric methods for digital picture analysis. Morgan Kaufmann Publishers, San Francisco, Amsterdam.

[18] Kong, T., Roscoe, A., Rosenfeld, A., 1992. Concepts of digital topology. Topology and its Applications 46 (3), 219 – 262.

[19] Kong, T. Y., Rosenfeld, A., 1989. Digital topology: Introduction and survey. Computer Vision, Graphics, and Image Processing 48 (3), 357–393.

[20] Latecki, L. J., 1997. 3D Well-Composed Pictures. CVGIP: Graphical Model and Image Processing 59 (3), 164–172.

[21] Lee, C., Poston, T., Rosenfeld, A., 1991. Winding and Euler numbers for 2D and 3D digital images. CVGIP: Graphical Model and Image Processing 53 (6), 522–537.

[22] Lee, C., Poston, T., Rosenfeld, A., 1993. Holes and Genus of 2D and 3D Digital Images. CVGIP: Graphical Model and Image Processing 55 (1), 20–47.

[23] Lin, X., Xiang, S., Gu, Y., 2008. A new approach to compute the Euler Number of 3D image. In: IEEE Conference on Industrial Electronics and Applications (ICIEA). pp. 1543–1546.

[24] Lobregt, S., Verbeek, P. W., Groen, F. C. A., 1980. Three-Dimensional Skeletonization: Principle and Algorithm. IEEE Trans. Pattern Anal. Mach. Intell. 2 (1), 75–77.

[25] Lozano-Durán, A., Borrell, G., 2016. Algorithm 964: An Efficient Algorithm to Compute the Genus of Discrete Surfaces and Applications to Turbulent Flows. ACM Trans. Math. Softw. 42 (4), 34:1–34:19.

[26] Maunder, C. R. F., 1980. Algebraic Topology. Cambridge University Press.

[27] McAndrew, A., Osborne, C., 1997. The Euler characteristic on the face-centred cubic lattice. Pattern Recognition Letters 18 (3), 229–237.

[28] Morgenthaler, D., 1980. Three-Dimensional Digital Topology: the Genus. Tech. Rep. TR-980, University of Maryland, College Park, MD 20742.

[29] O'Neill, B., 1966. Elementary Differential Geometry. Elsevier Inc.

[30] Park, C. M., Rosenfeld, A., 1971. Connectivity and genus in three dimensions. Tech. Rep. TR-156, University of Maryland Computer Science Center, College Park, Maryland.

[31] Saha, P. K., Chaudhuri, B. B., 1995. A new approach to computing the Euler characteristic. Pattern Recognition 28 (12), 1955–1963.

[32] Saha, P. K., Chaudhuri, B. B., 1996. 3D Digital Topology under Binary Transformation with Applications. Computer Vision and Image Understanding 63 (3), 418–429.

[33] Sánchez-Cruz, H., Sossa-Azuela, H., Braumann, U.-D., Bribiesca, E., 2013. The Euler-Poincaré Formula Through Contact Surfaces of Voxelized Objects. Journal of Applied Research and Technology 11, 65–78.

[34] Siqueira, M., Latecki, L. J., Tustison, N. J., Gallier, J. H., Gee, J. C., 2008. Topological Repairing of 3D Digital Images. Journal of Mathematical Imaging and Vision 30 (3), 249–274.

[35] Sossa, H., 2016. On the Computation of the Number of Bubbles and Tunnels of a 3-D Binary Object. In: Proceedings of the 5th International Conference on Pattern Recognition Applications and Methods, ICPRAM. pp. 17–23.

[36] Stelldinger, P., Latecki, L. J., Siqueira, M., 2007. Topological Equivalence between a 3D Object and the Reconstruction of its Digital Image. IEEE Trans. Pattern Anal. Mach. Intell. 29 (1), 126–140.

[37] Toriwaki, J., Yonekura, T., 2002. Euler Number and Connectivity Indexes of a Three Dimensional Digital Picture. Forma 17, 183–209.

[38] Voss, K., 1991. Images, objects and surfaces in Z^3. IJPRAI 5, 797–808.

[39] Ziou, D., Allili, M., 2002. Generating cubical complexes from image data and computation of the Euler number. Pattern Recognition 35 (12), 2833–2839.