Ann Math Artif Intell (2013) 67:71–80
DOI 10.1007/s10472-013-9336-z

A continuous analog for 4-dimensional objects

A. Pacheco · J.-L. Mari · P. Real

Published online: 8 March 2013

Abstract In this paper, we follow up on the studies developed by Kovalevsky (Comput Vis Graph Image Process 46:141–161, 1989) and Kenmochi et al. (Comput Vis Image Underst 71:281–293, 1998), which defined a continuous analog for a 4-dimensional digital object. Here, we construct a cell complex that has the same topological information as the original 4-dimensional digital object.

Keywords Binary object · Cell complex · Discrete cell · Embedded cell · Isometry

Mathematics Subject Classifications (2010) 52B11 · 52B55 · 68U05

1 Introduction

Many important notions of image analysis, such as the connectivity of regions, boundaries and adjacency, are important topics in topology.

The main difficulty in formalizing these notions and making them accessible for a computer is the following: whereas modern general topology considers there to be

This research has been funded by the Spanish Ministry of Science and Innovation 4D-Hom (reference: MTM2009-12716) and FEDER funds.

A. Pacheco (✉) · P. Real
Dpto. Matemática Aplicada I, E.T.S.I. Informática,
Universidad de Sevilla, Avda. Reina Mercedes, s/n 41012 Sevilla, Spain
e-mail: ampm@us.es

P. Real
e-mail: real@us.es

J.-L. Mari
Information and System Science Laboratory (LSIS, UMR CNRS 7296),
Aix-Marseille Université, Campus de Luminy, case 925,
13288 Marseille Cedex 9, France
e-mail: jean-luc.mari@univ-amu.fr

infinitely many points in an arbitrary neighborhood of a point, a digital object has a finite number of elements by necessity.

One way to deal with this issue is to work within a discrete context. In order to correctly retrieve some topological notions and properties, it is necessary to define a continuous analog for digital objects.

Several authors have proposed different solutions to the problem of embedding a digital object into a continuous representation (see [5, 20]). At the end of the 1980s, V. A. Kovalevsky looked for the solution to this problem in the classical general topology (see [14]). After analyzing the attempts of imperfect solutions that were proposed by other authors, Kovalevsky noticed that most of them had tried to consider the digital plane as a structure that consists of elements that have different dimensions. Such a structure is well-known in the literature, and it is called a *cell complex*. Moreover, Kovalevsky proved that the concept of a cell complex removes the topological paradoxes and contradictions in the theory of 2-dimensional digital objects. In addition, he suggested that this concept can be applied without any change to describe the topological structure of n-dimensional digital objects.

The procedure that is used to encode digital objects in cellular terms is based on the consideration that such objects can be defined by a map from the set of n-xels to a set of labels. More concretely, the neighboring n-xels that have the same assigned label are contained in the same region, i.e., in the same n-dimensional cell subcomplex. By combining these subcomplexes, a cell complex that defines a continuous analog for the object and allows us to retrieve topological notions can be constructed.

There already exist several methods that construct complexes from lower-dimensional digital images (see, for example, [9, 15] and [11–13]). These methods construct complexes that represent the topology of the region of interest in a binary digital image. More concretely, in [9, 15], the algorithm constructs a simplicial complex whose 0-cells are points of the edges of the sub-grid. Similarly, in [11–13], the authors construct a cell complex on a dual grid, i.e., they construct a cell complex whose 0-cells are vertices of the sub-grid.

Although previous methods have worked with digital images of at most dimension 3, they have not examined digital images of higher-dimension. Here, we extend the current techniques to associate complexes with digital images of higher dimension. More precisely, we work with digital images of the lowest non-studied dimension, i.e., we deal with 4-dimensional digital images.

In order to represent digital objects using computational techniques, it is necessary to fix both a grid and the relations between the points. We consider binary objects that have been derived from a subdivision of 4-dimensional space into unit hypercubes, which intersect two by two in a cubic face. This grid is an extension of the grid that is used in [17] for $n = 3$. Equivalently, we could use the 4-dimensional discrete space $\mathbb{Z}^4$ as the grid, in which case the lattice points would be the elements $(i, j, k, l) \in \mathbb{Z}^4$. Once the grid has been established, we must fix the neighborhood relations between the lattice points. For a given lattice point, a neighborhood is typically defined using a distance metric (see [4]). In this paper, we use the 80-neighborhood in $\mathbb{Z}^4$, i.e., given a point $P = (i, j, k, l)$ in $\mathbb{Z}^4$, the neighbors of P are the points $Q = (i', j', k', l')$ that satisfy $d_1(P, Q) = |i - i'| + |j - j'| + |k - k'| + |l - l'| \leq 4$ and $d_\infty(P, Q) = max\{|i - i'|, |j - j'|, |k - k'|, |l - l'|\} = 1$ (see [8] for more details).

We consider the central point of each hypercube of the grid and we construct a sub-grid that is composed of hypercubes whose vertices are the central points of

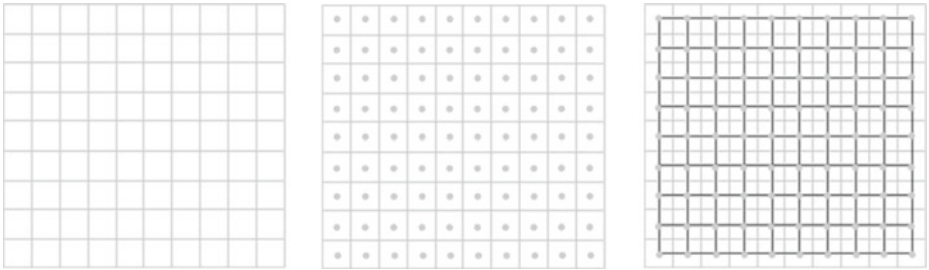

Fig. 1 We consider the central point of each unit square of the 2-dimensional grid and we construct a sub-grid that consists of squares whose vertices are the central points of the squares of the original grid

the hypercubes of the original grid. In this way, the 4-xels of a 4-dimensional object are identified with the vertices of the hypercubes of the sub-grid. See Fig. 1 for a 2-dimensional example.

In this paper, we describe a method to convert the hypercubes of the sub-grid into the blocks of the cell complex that defines a continuous analog for a given binary object. More concretely, we construct a hyperpolyhedron that approximates a given digital object as closely as possible. This construction is made by attaching cells to obtain larger cells. The complex represents a partition of an image into regions and it indicates the incidence and adjacency relations between regions. In the case of binary images, there exists only one region of interest; thus, it suffices to represent this boundary region's.

The structure of this paper is as follows: (a) the vertices of the unit hypercubes of the sub-grid that correspond to 4-xels of the object are determined (Section 3); (b) the convex hull of these vertices is constructed, and as a result, the blocks of the complex are obtained (Section 4); finally, (c) we present some conclusions about the method shown here (Section 5).

2 Preliminaries

In order to enhance the comprehensibility of this paper to the general reader, we introduce some useful concepts, e.g., a cell complex, an isometry, and a hypercube.

A *q-dimensional cell* (or *q-cell*) is a topological space that is homeomorphic to an open ball $\mathbb{B}^q(x_0, r) = \{x \in \mathbb{R}^q : d(x, x_0) < r\}$. Let σ be a q-cell. Then, the boundary of σ is defined as $\partial\sigma = cl(\sigma)\backslash\sigma$, where $cl(\sigma)$ is the closure of σ. Let us note that $\partial\sigma \neq \emptyset$, as σ is an open set. Moreover, we say that a $(q-1)$-dimensional cell τ bounds σ if $\tau \neq \sigma$ and if $cl(\tau) \subset cl(\sigma)$, where $cl(\tau)$ is the closure of τ. The $(q-1)$-dimensional cell τ is called the *face* of σ.

A *cell complex* is a set $K = \{K_q\}_{q\geq 0}$ of cells that satisfy two conditions: (1) every face of a cell is a cell and (2) if σ and σ' are cells, then their intersection is either a common face of both cells or it is empty. Moreover, if the intersection of two cells is a common face, then this face is present twice. Because we work in the field $\mathbb{Z}_2$, we know that $1 + 1 = 0$, hence, the face vanishes. For more details about cell complexes see [6].

An object $O \subseteq \mathbb{R}^n$ is called *full-dimensional* if O is n-dimensional.

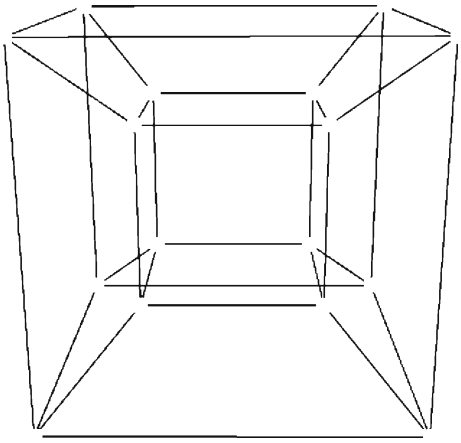

Fig. 2 A Schlegel diagram of a hypercube

A map $f : X \to Y$ is called an *isometry* if for any $a, b \in X, d(a, b) = d(f(a), f(b))$. Two objects O, O' are called *isometric* (or *congruent by isometries*) if there exists a bijective isometry from O to O'.

A *n-polytope* is the convex hull of a finite subset of points that generate[1] a space of dimension n. In particular, a *polygon* is the convex hull of a subset of points in a plane, i.e., a 2-polytope; similarly, a *polyhedron* is the convex hull of a subset of points in space, i.e., a 3-polytope.

A *hypercube* is the convex hull of sixteen vertices in 4-dimensional space, i.e., a 4-polytope. Moreover, each vertex is a distance of one unit from four vertices, a distance of two units from six vertices, a distance of three units from four vertices, and a distance of four units from one vertex.

A *Schlegel diagram* is a projection of an n-polytope from $\mathbb{R}^n$ into $\mathbb{R}^{n-1}$ from a point that is beyond one of its faces.

In Fig. 2, we show a Schlegel diagram of a hypercube. This representation is used in this paper.

The unit hypercube $H = L_1 \times L_2 \times L_3 \times L_4$ (with $L_1 = L_2 = L_3 = L_4 = [0, 1]$) can be seen as a 4-dimensional cell complex whose 0-cells, 1-cells, 2-cells, 3-cells, and 4-cells are their vertices, edges, square faces, cubic faces, and the hypercube itself, respectively.

The *group of isometries of a hypercube* are the rigid motions that leave the hypercube invariant. This group has 384 elements (see [3] for more details).

3 The determination of discrete cells

In this section, we determine the non-isometric finite subsets of points that can be constructed by starting from the vertices of the hypercubes that subdivide the discrete space $\mathbb{Z}^4$. The points of each of these finite subsets (called the point set and denoted by V) are the vertices of a *discrete cell* in $\mathbb{Z}^4$.

[1] Observe that a k-dimensional space is generated by $k + 1$ linearly independent points.

We assign a binary value of zero or one to each lattice point in $\mathbb{Z}^4$. In addition, we assume that the points of V have a value of 1 and that the points in the complement of V have a value of 0. The points that have a value of 1 (respectively, 0) are called 1-points (respectively, 0-points). Because a hypercube has 16 vertices and because each vertex can be a 1-point or a 0-point, there are $2^{16} = 65536$ finite subsets of points. More concretely, there are $C(16, c)$ subsets with c 1-points of the hypercube. By considering the subsets' complements, it is clear that $C(16, 16 - c) = C(16, c)$ is also the number of subsets with $16 - c$ 1-points.

The developed method determines the non-isometric finite subsets of points that can be constructed by starting from the vertices of a unit hypercube. This method consists of computing the group of isometries of the unit hypercube in $\mathbb{R}^4$ and using this group to ignore the subsets of points that differ by the isometries of 4-dimensional space. Proposition 7 in [21] proves that an algorithm of this type returns the discrete cells in $\mathbb{Z}^4$. A scheme of this algorithm is shown in Fig. 3.

Remark 1 **Automorphisms[Hypercube[4]]** in the *Mathematica* package **Combinatorica'** gives a list with the 384 permutations of vertices that leave the unit hypercube invariant. Each of these permutations corresponds to an isometry of the unit hypercube.

When we have obtained the group of the isometries of a hypercube, we implement Algorithm 1 to determine the discrete cells. This algorithm constructs an ordered set $(V_c, \prec)$ that contains the $C(16, c)$ subsets with $0 \leq c \leq 16$ vertices of the unit hypercube. Next, for $(V_c)_i \subset V_c$, the algorithm computes all of the subsets that are isometric to $(V_c)_i$ by applying the group of isometries of the hypercube. If there exists a subset $(V_c)_j \subset V_c$ that coincides with any of the subsets that are isometric to $(V_c)_i$, then the algorithm removes $(V_c)_j$ from (V_c).

Note that in [18], an algorithm (namely Algorithm 1) that is based on graphs was presented to compute the non-congruent finite subsets that are obtained by

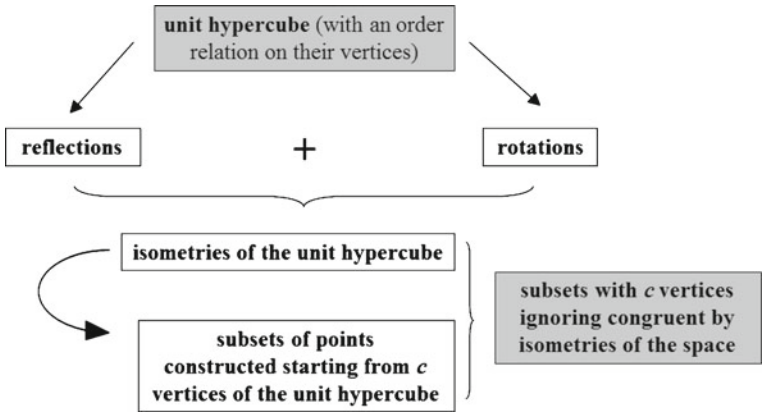

Fig. 3 A scheme of an efficient algorithm that ignores those subsets of vertices of the unit hypercube that differ by the isometries of 4-dimensional space. The input and output data are shaded in *gray*

Algorithm 1

Input: $(V, \prec)$ is the set of vertices of the unit hypercube that is arranged using the order relation $\prec$.

iso_hypercube: the group of isometries of the unit hypercube.

Output: non-isometric subsets with $0 \leq c \leq 16$ points.

begin

// *NIS*: an empty list that saves the non-isometric subsets with $0 \leq c \leq 16$ points

for $c = 0, ..., 16$ **do**

Construct the ordered set $(V_c, \prec)$ that contains each of the $C(16, c)$ subsets with exactly c points

for $(V_c)_i \subset V_c$ **do**

$iso\_vci = \emptyset$

{An empty list that is used to save the subsets that are isometric to $(V_c)_i$.}

for $\sigma \in iso\_hypercube$ **do**

$iso\_vci = iso\_vci \bigcup \sigma((V_c)_i)$

end for

for all $(V_c)_j \subset V_c$ satisfying $(V_c)_j \in iso\_vci$ **do**

{$(V_c)_i$ and $(V_c)_j$ are isometric}.

$V_c = V_c - \{(V_c)_j\}$

end for

end for

$NIS = NIS \bigcup V_c$

end for

return *NIS*

end

the isometries of 4-dimensional space. This algorithm had a high computational complexity because the graph isomorphism problem is NP-complete (see [2, 7] for more information). In this sense, the algorithm whose scheme is shown in Fig. 3

Table 1 The number of discrete cells and the computational times required by Algorithm 1 in [14] and Algorithm 1, respectively; these algorithms were implemented in Mathematica 7.0.0 using AMD Turion(tm) 64 X2 Mobile Technology TL-58 1.90 GHz

Vertices	0	1	2	3	4	5
Cells	1	1	4	6	19	27
Algorithm 1 in [14]	NA	NA	NA	NA	NA	NA
Algorithm 1	NN	NN	5.67 sec	9.54 sec	31.32 sec	53.1 sec

Vertices	6	7	8	9	10	11
Cells	50	56	74	56	50	27
Algorithm 1 in [14]	NA	NA	11092.3 sec	9482.27 sec	11652.5 sec	8382.22 sec
Algorithm 1	115.08 sec	149.55 sec	223.39 sec	189.3 sec	186.73 sec	112.68 sec

Vertices	12	13	14	15	16
Cells	19	6	4	1	1
Algorithm 1 in [14]	2384.01 sec	734.34 sec	175.04 sec	25.78 sec	NN
Algorithm 1	84.7 sec	29.85 sec	21.26 sec	6.02 sec	NN

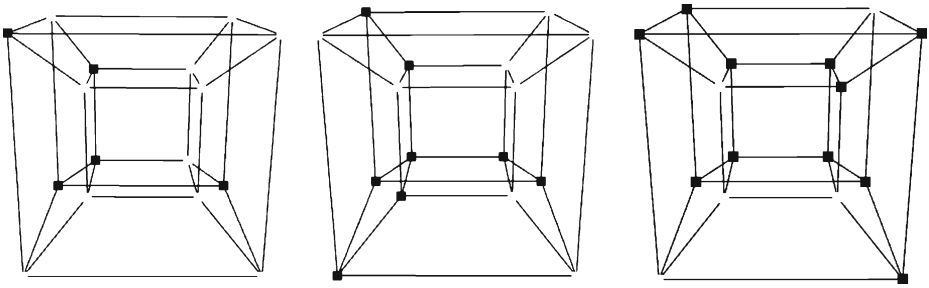

Fig. 4 From *left* to *right*: a discrete cell with five vertices, a discrete cell with eight vertices and a discrete cell with 11 vertices

improves the computational time of Algorithm 1. In Table 1, we present results that were obtained by using both algorithms.[2]

In Fig. 4, we show examples of some of the discrete cells that are presented in Table 1.

4 The determination of the embedded cells

The goal of this section is to embed the discrete cells in $\mathbb{R}^4$. The *embedded cell EC* that is associated with the discrete cell *DC* is defined as the set of points that includes all of the points inside of the convex hull of the vertices of *DC*.

In order to determine these convex hulls, (1) we distinguish four basic cases that consist of joining each pair of vertices of the discrete cell and (2) we define an algorithm (Algorithm 2) that is based on constructing 3-dimensional convex hulls and attaching the remaining edges and faces.

Basic cases

- If the discrete cell (whose convex hull must be computed) has only two vertices, we attach an edge that connects this pair of vertices. The corresponding embedded cell is an edge.
- If the discrete cell has three vertices, we attach edges that connect each vertex to the other two vertices. Thus, the embedded cell is a triangle.
- If the discrete cell has four non co-planar vertices, we attach edges that connect each vertex to the other three vertices. Hence, the embedded cell is a tetrahedron.
- Similarly, if the discrete cell has five non co-spatial vertices, we attach edges that connect each vertex to the other four vertices, and the embedded cell is a hypertetrahedron.

Remark 2 Note that the previous basic cases correspond to cells whose vertices are each connected to all of the other vertices by an edge.

[2]NA: this algorithm is not applicable to cells with that number of vertices. NN: it is not necessary to compute the cells with that number of vertices.

Algorithm 2

Input: the discrete cell DC.
Output: the embedded cell EC that is associated with DC.
begin
// E: an empty set that saves the edges of EC
  for each of the eight cubes C_i of the hypercube H **do**
    Determine the set E_i of edges of the convex hull of the points of DC that are contained in C_i
    $E = E \bigcup E_i$
  end for
  Determine the set F of the faces of the convex hull from the set E of edges
  Determine the set V of the volumes of the convex hull from the set E of edges
  for each non-planar face $f \in F$ **do**
    Attach an edge e that bisects f into two planar faces f_1, f_2
    $E = E \bigcup \{e\}$
    $F = F - \{f\} \bigcup \{f_1\} \bigcup \{f_2\}$
  end for
  return E,F,V
end

Note that every set E_i of edges is obtained by applying a set of integral operators to each cube C_i (see [19]). Moreover, the set F (respectively, V) of faces (respectively, volumes) is obtained by constructing edges that do not cross and triangles (respectively, tetrahedra) by using 3-cycles (respectively, 4-cycles) of edges (see [10]).

First, Algorithm 2 computes the set of edges of the convex hull of each subset of vertices that is contained in a cubic face of the hypercube. This set is obtained using the procedure described in [19]. Then, Algorithm 2 determines the sets of faces and volumes of this convex hull. These sets are determined using the method given in [10]. Finally, Algorithm 2 attaches (in the cases where it is necessary) an inner edge of the hypercube and the corresponding faces in order to obtain the 4-dimensional convex hull.

The embedded cells in $\mathbb{R}^4$ can be obtained using Algorithm 2. In particular, by using the 402 discrete cells as input data, we obtain the 402 embedded cells in $\mathbb{R}^4$. In Fig. 5, we show the embedded cells that are associated with the discrete cells shown in Fig. 4.

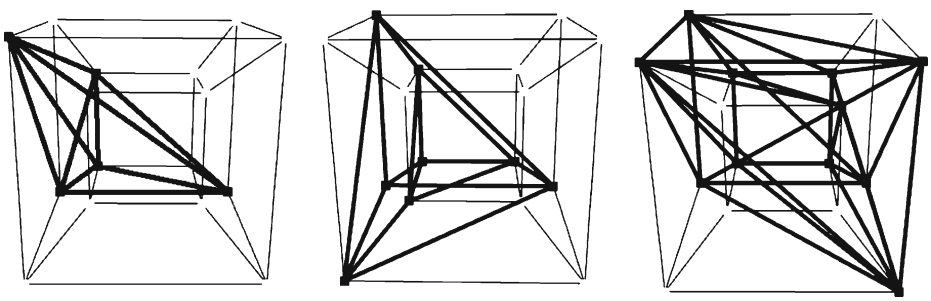

Fig. 5 The embedded cells that are associated with the discrete cells shown in Fig. 4

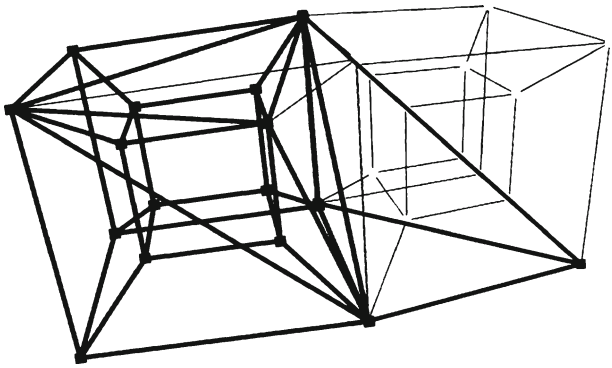

Fig. 6 A cell complex associated with a 4-dimensional binary digital image that consists of 16 *black points*

5 Conclusions

In this paper, we have shown a computational method that (1) determines the discrete cells in the space $\mathbb{Z}^4$ and (2) embeds these cells into $\mathbb{R}^4$. By running Algorithm 1 in the symbolic computation package Mathematica, 402 discrete cells contained in the unit hypercube were obtained. These 402 discrete cells are embedded in the space $\mathbb{R}^4$ and the convex hulls are computed using Algorithm 2. Note that 347 of these embedded cells are full-dimensional cells of dimension 4 (cf. the second column of Table 3 in [1]). These 402 embedded cells are saved in a look-up table. By combining the cells of the look-up table, a continuous convex analog for a given 4-dimensional digital object can be constructed. More concretely, the procedure for constructing the cell complex consists of these three steps: (1) associate each of the points of the image with a discrete cell; (2) construct the embedded cells from the discrete cells; and (3) invert the isometry between each subset of points of the image and its associated discrete cell. In Fig. 6, we show an example of a 4-dimensional cell complex that is constructed by attaching two cells.

The procedure developed in this paper can be generalized to higher dimensions. Moreover, (1) to determine the discrete cells in nD, it suffices to compute the group of isometries of a hypercube of dimension n; similarly, (2) to construct the embedded cells, it suffices to determine the convex hull of these discrete cells.

Finally, advanced topological information (e.g., cohomology algebra, homology, A(∞)-coalgebras, and (co)homology operations) can be derived directly from this continuous cell analog in a similar way to the method described in [16].

References

1. Aichholzer, O.: Extremal properties of 0/1-polytopes. Oberwolfach Semin. **9**, 111–130 (2000)
2. Arias-Fisteus, J., Fernández-García, N., Sánchez-Fernández, L., Delgado-Kloos, C.: Hashing and canonicalizing notation 3 graphs. J. Comput. Syst. Sci. **76**(7), 663–685 (2010)
3. Costa, S.R., Gerônimo, J.R., Palazzo, R. Jr., Carmelo Interlando, J., Muniz Silva Alves, M.: The symmetry group of $\mathbb{Z}_q^n$ in the Lee space and the $\mathbb{Z}_{q^n}$-linear codes. Lect. Notes Comput. Sci. **1255**, 66–77 (1997)
4. Dörksen-Reiter, H.: Shape representations of digital sets based on convexity properties. Dissertation, Universität Hamburg (2005)

5. Feng, H.J., Pavlidis, T.: The generation of polygonal outlines of objects from gray level pictures. IEEE Trans. Circuits Syst. **22**(5), 427–439 (1975)
6. Fritsch, R., Piccinini, R.A.: Cellular Structures in Topology. Cambridge University Press, Cambridge (1990)
7. Gross, J.L., Yellen, J.: Handbook of Graph Theory. CRC Press (2003)
8. Han, S.E.: A generalized digital (k_0, k_1)-homeomorphism. Note Mat. **22**(2), 157–166 (2003)
9. Hanisch, F.: Marching square. CGEMS: Computer graphics educational materials source (2008)
10. Jesson, N., Mari, J.-L., Molina-Abril, H., Real, P.: S implicialization of octrees. In: Proceedings of the First Workshop on Computational Topology in Image Context, pp. 1–6 (2008)
11. Kenmochi, Y., Imiya, A.: Combinatorial boundary of a 3D lattice point set. Vis. Commun. Image Represent. **17**(4), 738–766 (2006)
12. Kenmochi, Y., Imiya, A., Ezquerra, N.F.: Polyhedra generation from lattice points. In: Proceedings of Discrete Geometry for Computer Imagery. Lecture Notes in Computer Science, vol. 1176, pp. 127–138 (1996)
13. Kenmochi, Y., Imiya, A., Ichikawa, A.: Boundary extraction of discrete objects. Comput. Vis. Image Underst. **71**, 281–293 (1998)
14. Kovalevsky, V.: Finite topology as applied to image analysis. Comput. Vis. Graph. Image Process. **46**, 141–161 (1989)
15. Lorensen, W.E., Cline, H.E.: Marching cubes: a high-resolution 3D surface construction algorithm. Comput. Graph. **21**(4), 163–169 (1987)
16. Mari, J.-L., Real, P.: Simplicialization of digital volumes in 26-adjacency: application to topological analysis. Pattern Recogn. Image Anal. **19**(2), 231–238 (2009)
17. Pacheco, A., Real, P.: Polyhedrization, Homology and Orientation, pp. 151–164. Research Publishing Services (2009)
18. Pacheco, A., Real, P.: Associating cell complexes to four dimensional digital objects. In: Proceedings of Discrete Geometry for Computer Imagery. Lecture Notes in Computer Science, vol. 6607, pp. 104–115 (2011)
19. Pacheco, A., Real, P.: Determining Bricks of 3–Dimensional Cell Complexes, pp. 37–48. Research Publishing Services (2011)
20. Rosenfeld, A.: Connectivity in digital pictures. J. ACM **17**(1), 146–160 (1970)
21. Ziegler, G.M.: Lectures on 0/1-polytopes. Oberwolfach Semin. **29**, 1–41 (2000)