Linear algebraic representation of geometric data

Antonio DiCarlo *Alberto Paoluzzi* *Vadim Shapiro*

October 7, 2012

Contents

1 Introduction

The challenge we face up "it does matter to extract as much topological and metrical information as possible from raw 3D tomographic data" [2]

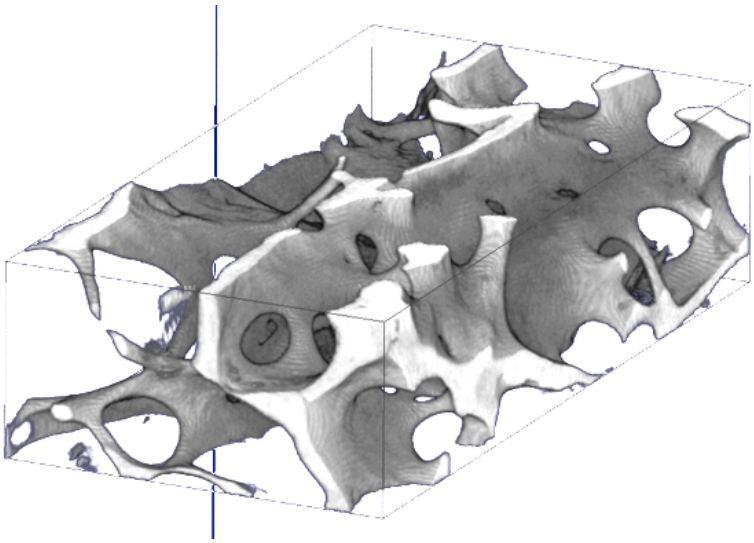


Figure 1: A volume rendering of trabecular geometry extracted [2] from a tomographic images of a (small) portion of spongy bone.

1.1 Representation scheme

The notion of *representation scheme* mapping $s : M \to R$ from a space of math models M to computer representations R.

1. The set M contains the mathematical models of the class of solid objects that the scheme aims to represent.

2. The set R contains symbolic representations, i.e. suitable data structures, built according to some appropriate computer grammar.

1.2 Previous work

The origins The extremely rich literature about *schemes of representation* in solid modeling, that we synthetize here without any tentative of

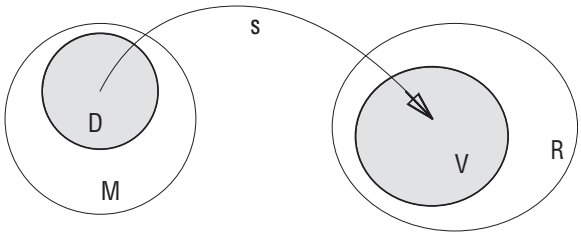


Figure 2: A representation scheme s as a mapping from a domain D contained in a space of mathematical models, to a set of valid representations in a space of computer representation generated by a given grammar [23, 26].

completeness, started with the foundational paper [23] by Requicha and the PAP (Project Automation Project) directed by Voelcker at the University of Rochester in late 1970s. A mathematical framework for characterising the important aspects of *computer representations of solids* was introduced, and used to describe and compare all of the major schemes for representing solids known at the time. The Baumgart's ground-breaking paper [3] had previously supplied the first but already efficient *boundary representation* scheme, afterwards used as a unit of measure for comparing later schemes. It had also introduced the so-called *Euler operators*, as elementary operations for step-wise building a well-formed polyhedron, using atomic software actions that satisfy in every state the Euler-Poincaré formula. A hierarchical boundary representation made by positive or negative bodies, bounded by faces living on a surface and limited by one or more cycles of possibly curved edges, in turn limited by two 3D vertices, was introduced by Ian Braid in [5]. Only manifold solids could by represented and composed, via efficient algorithms to built new objects by adding and subtracting simpler objects.

Boundary representations The *Quad-Edge* data structure, that provided efficient primitives for manipulation of planar subdivisions and the computation of Voronoi diagrams was proposed in [10], and is yet in large use in computational geometry algorithms and in geometric libraries. A decomposition of a polyhedron into tetrahedra, and hence the modeling of polyhedral complexes in R^3 and surfaces of 4-polyhedra, was made possible by the [8] *Facet-Edge* data structure, that can be considered an higher dimensional generalisation of [10]. A boundary representation for manifold topology using Euler operators and the *Half-Edge* data structure, together with a C++ library implementing the core functionality of a solid modelling

system was given in [16]. In [13] the representation and manipulation of points, lines, polygons, and solids was integrated in a single model, using the *Hybrid-Edge* data structure and the related operations based on Euler operators, in order to facilitate the design of artefacts at many levels of design abstraction. The set of models representable in the scheme [18] is a Boolean algebra that contains the disconnected, nonconvex, unbounded and nonmanifold linear polyhedra. The range set of the representation scheme is the subset of regular closed 2-complexes. The associated data structure, named *Winged-Triangle*, is space optimal for linearised representations of polyhedra with curved boundary. The storage complexity of boundary data structures was studied by [29] together with the time complexity of a set of primitive queries and update routines, and by analysing the nine binary relationships between the three main boundary entities F (faces), E (edges), and V (vertices). In [14] a set of generalised Euler operators are provided to manage a non-manifold boundary representation, called the *Partial-Entity* data structure, through partial topological entities such as the partial-face, partial-edge, and partial-vertex, used for describing non-manifold conditions at vertices, edges, and faces. The efficiency of out-of-core access to geometric data describing boundary representation in a CIM environment was studied by [1], showing an excellent storage and retrieval efficiency for the *Delta* data structure. In [15] some of the most representative ordered topological models are compared using quite intricate concepts of *darts*, *n-dimensional generalised maps* and *n-dimensional maps*, used to study subdivisions of orientable or non-orientable topological spaces, and subdivisions of oriented topological spaces, respectively.

Cellular decompositions In [25], the focus was on *dimensionally non-homogeneous, non-closed pointsets with internal structures*, that may deal with mixed-dimensional objects with dangling or missing boundary elements. More general set-theoretic and topological operators are provided to represent inhomogeneous objects. Euler operators with boundary and neighborhood constraints on both bounding entities (vertices, edges, faces, loops, shells, and regions) and coupling entities (ends, sides, corners, and feathers) were used by [30] to give boundary representations with *non-manifold topology*. With respect to topology, geometry and structure, the *hierarchical representation* scheme proposed by [19] could be seen exactly half-way between solid modeling and contemporary standard graphics, since an acyclic multigraph (reminiscent of *scene graphs*) was associated to any model, in order to store the hierarchical object structure, including the operations and

sub-components which generate any object part. The *Selective Geometric Complex* (SGC), a generalisation of *CW-complexes* allowing for cells non homeomorphic to open discs, was proposed by [24] to handle dimension-independent models of point sets with internal structures and incomplete boundaries. Four different data structures for the representation of topological information for curved surface and solid modeling are discussed in [28], exploring the time, space, and complexity requirements of each. In [6] the CW-complex is provided as the abstract framework for numerical representation of dimensionally non-homogeneous pointsets with internal structure. In [17] a simplicial-based dimension-independent representation is presented, and various operators and algorithms are discussed, including boundary evaluation, linear and screw extrusion, grid generation, simplicial maps, and set operations.

Recent approaches The article [26] reviewed the main ideas and foundations of solid modeling in engineering. In particular, it remarked that solid modeling was conceived as a *universal technology* for developing engineering languages and systems with guaranteed geometric validity, where the representation should support all geometric queries that may be asked about the corresponding physical object. A Shape Modelling API for the STEP Standard was given in [20], extending the standard to enable the capture and transfer of *parametrized CAD models* with g*eometric constraints*, even in procedural form, i.e. expressed in terms of the sequence of operations used to construct them. The Djinn application programmers interface (API) for solid modelling [4] aimed to specify only the *modelling abstractions* useful at user-level, leaving that the implementations might employ either the boundary representation or a set-theoretic paradigm, or any other representation that unambiguously defines solid objects. Reference [9] introduced a general mathematical theory of n-dimensional geometric objects that allows boundary representations as a special case, by combining *sub-analytic geometry*, where pointsets are defined in a way broader than semi-analytic sets, and Whitney's *theory of stratifications*. The notion of *parametric family* of solids is defined and elucidated by [27], considering that the high productivity of parametric systems lie in the ability to allow later edits by changing a few input parameters that were specified during the creation of the design. A formal definition of *boundary representation (BR-)deformation* for solids in the same parametric family, including necessary conditions that must be satisfied by any BR-deforming mappings, applicable to more general cellular models of pointsets, is given in [21]. In [12] it is addressed the hard problem

of *validity of parametric CAD models*, giving an algorithm that computes for planar parametric models the valid parameter ranges within which the model will regenerate. The necessary and sufficient conditions for consistency of the parametric and boundary representation after updates applied to either the parametric model and/or to the boundary representation are given by [22], relying on *sign-invariant space decompositions*. In [7] it is shown that the (co)chain complex associated with a decomposition of the computational domain, commonly called a *mesh* in computational science and engineering, can be represented by a sparse block-bidiagonal matrix whose blocks codify the incidence and adjacency relations between cells of various dimensions, and represent the *boundary* and *coboundary operators* of the *chain complex* supported by the mesh.

2 Proposal

Here we introduce a novel *linear algebraic representation* (LAR) scheme for a large class of geometric shapes in $\mathbb{E}^n$. The goal is to provide a representation that should support (at least in principle) any and all geometric queries that may be asked of the corresponding physical object.

2.1 Linear Algebraic Representation (LAR scheme)

It is worth to think that the descriptive power of a scheme is measured by the size of its domain. In this sense, the LAR scheme — based on chains of cells — introduced in this paper is defined on a quite large mathematical domain, and may represent a large class of cell decompositions, including *simplicial*, *cuboidal*, *polytopal*, and *polyhedral complexes*, and much more. It is also safe to denote it as a *linear scheme*, because the scalars of the chain combinations are taken from a field[1].

Definition 1 (Math models)**.** *The LAR scheme is defined on the pairs* $(\mathcal{S}, \mathcal{C})$*, where* $\mathcal{S}$ *is a set of* finite CW-complexes*, and* $\mathcal{C}$ *is a class of* characteristic maps*.*

Definition 2 (Computer representations)**.** *The LAR scheme takes values into the set* R *of sparse matrix representations*[2] *of a* chain complex *of* singular p-chains $(0 \leq p \leq d)$*.*

[1] $\mathbb{Z}_2 = \mathbb{Z}/2\mathbb{Z} = \{0, 1\}$ *is* a field.

[2] Typically using a CSR (Compressed Sparse Row) format.

Remark For the sake of simplicity, the LAR scheme is restricted in this paper to the set $\mathcal{S}$ of regular simplicial complexes and to the set $\mathcal{C}$ of affine simplicial maps. Analogously, the representations R are given by the CSR matrices of linear operators between groups of chains/cochains with coefficients either in $\mathbb{Z}_2 = \{0, 1\}$, or in $\{-1, 0, 1\}$.

Disks and spheres The *n-disk* and the *n-sphere* are the sets $D^n = \{x \in \mathbb{R}^n : |x| \leq 1\}$ and $S^n = \{x \in \mathbb{R}^n : |x| = 1\}$, respectively.

Definition 3 (Cell)**.** *An* n-cell *is a space homeomorphic to the open* n*-disk* $\text{int}(D^n)$*. A* cell *is a space which is an* n*-cell for some* $n \geq 0$*.*

Definition 4 (Cell-decomposition)**.** *A* cell-decomposition *of a space* X *is a family*

$$\mathcal{E} = \{e_\alpha | \alpha \in I\}$$

of subspaces of X *such that each* e_α *is a cell and* $X = \bigsqcup_{\alpha \in I} e_\alpha$ *(disjoint union of sets).*

Definition 5 (n-skeleton)**.** *The* n-skeleton *of* X *is the subspace*

$$X^n = \bigsqcup_{\alpha \in I:\ \dim(e_\alpha) \leq n} e_\alpha.$$

A *finite cell-decomposition* is a cell decomposition consisting of finitely many cells.

Definition 6 (Finite CW-complex)**.** *The pair* $(X, \mathcal{E})$*,where* X *is a Hausdorff space and* $\mathcal{E}$ *is a finite cell-decomposition of* X*, is called a* finite CW-complex *if for each* n*-cell* $e \in \mathcal{E}$ *there is a* characteristic map $\phi_e : D^n \to X$ *restricting to a homeomorphism* $\phi_{e|\text{int}(D^n)} : \text{int}(D^n) \to e$ *and taking* S^{n-1} *into* X^{n-1}*.*

2.1.1 Singular p-chains

Let G be any Abelian (i.e. commutative) group G. Groups of interest are

1. $G = \mathbb{Z}$, the group of integers,
2. $G = \mathbb{R}$, the additive group of reals,
3. $G = \mathbb{Z}_2 = \mathbb{Z}/2\mathbb{Z}$, the group of integers mod 2.

Definition 7 (Singular complex)**.** *A generalization of the simplicial complex can be defined on any manifold M or Hausdorff space X. For a* singular complex*, each singular simplex is a homeomorphism from a (Euclidean) simplex Δ_p in $\mathbb{R}^p$ to a subset of X.*

Definition 8 (Singular p-chain)**.** *Singular p-chain on a space X, with coefficients in G, is a* finite formal sum

$$c_p = g_1\sigma_p^1 + g_2\sigma_p^2 + \cdots + g_r\sigma_p^r$$

of singular simplexes $\sigma_p^s : \Delta_p \to X$*,each with coefficient $g_s \in G$, where Δ_p is the standard (Euclidean) p-simplex in $\mathbb{R}^p$.*

2.1.2 Chain and cochain complex

A solid representation based on chains and cochains applies to most domains characterized as cell complexes, without any restrictions on their type, dimension, codimension, orientability, manifoldness, and connectedness [7].

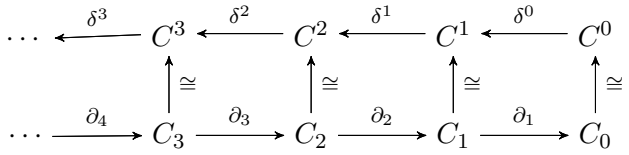


Figure 3: A (co)chain complex is a sequence of maps between Abelian groups or linear spaces.

Definition 9 (Chain complex)**.** *Chain complex is a sequence of Abelian groups $\ldots, C_3, C_2, C_1, C_0$ connected by homomorphisms (*boundary operators*) $\partial_n : C_n \to C_{n-1}$, such that for all n:*

$$\partial_{n-1} \circ \partial_n = 0$$

Definition 10 (Cochain complex)**.** *Cochain complex is a sequence of Abelian groups $C^0, C^1, C^2, C^3, \ldots$ connected by homomorphisms (*coboundary operators*) $\delta^n : C^n \to C^{n+1}$, such that for all n:*

$$\delta^{n+1} \circ \delta^n = 0$$

3 Some algorithms

Only a few fundamental algorithms associated with the LAR scheme are discussed in this preliminary essay. In the next papers we aim to provide generalisations to larger classes of cell decompositions and characteristic maps (including polynomial and rational maps), as well as to give other fundamental algorithms (including Boolean operations, Cartesian product and domain integration).

3.1 Facets extraction $C_p \to C_{p-1}$

In this section we discuss the extraction of the $(p-1)$-chain embedded within a p-chain. In particular, we discuss here the case of p-chains supported by a simplicial complex. Different extractions algorithms are required for different classes of cellular complexes.

Combinatorial formula It is well-known (see e.g. [17] or [11]) that the oriented facets [i.e. the $(p-1)$-faces] of a p-simplex $\sigma_p =< \sigma_0^0, \ldots, \sigma_0^p >$ are given by the simple combinatorial formula

$$\sigma_{p-1}^h = (-1)^h < \sigma_0^0, \ldots, \sigma_0^{h-1}, \sigma_0^{h+1}, \ldots, \sigma_0^p >, \qquad 0 \leq h \leq p \qquad (1)$$

In other words, each facet σ_{p-1}^h $(0 \leq h \leq p)$ of a simplex σ_p is computed by *dropping a vertex* from σ_p.

Matrix generation The formula (1) is translated here in matrix form, consistently with the algebraic approach adopted in this paper. The transformation from the input p-chains to the output $(p-1)$-chains is performed by the operator

$$\phi : C_p \to C_{p-1}; \quad M_p \mapsto M_{p-1}.$$

As we have already seen, any chain is represented with a CSR matrix. In few words, *every row* of an input matrix M_p generates $p+1$ (tentative) rows of the output matrix M_{p-1}. Duplicate rows are eliminated by sorting and removing the double rows.

$$M_2 = \begin{pmatrix} 0 & 1 & 3 \\ 1 & 2 & 4 \\ 1 & 3 & 4 \\ 2 & 4 & 5 \end{pmatrix} \quad \rightarrow \quad \begin{pmatrix} 1 & 3 \\ 0 & 3 \\ 0 & 1 \\ \\ 2 & 4 \\ 1 & 4 \\ 1 & 2 \\ \\ 3 & 4 \\ 1 & 4 \\ 1 & 3 \\ \\ 4 & 5 \\ 2 & 5 \\ 2 & 4 \end{pmatrix} \quad \rightarrow \quad M_1 = \phi(M_2) = \begin{pmatrix} 0 & 1 \\ 0 & 3 \\ 1 & 2 \\ 1 & 3 \\ 1 & 4 \\ 2 & 4 \\ 2 & 5 \\ 3 & 4 \\ 4 & 5 \end{pmatrix}$$

The above matrix M_2 gives a CSR representation of the binary characteristic matrix of the simplicial 2-complex shown in Figure 4. The matrix M_1 is a representation of the 1-chain embedded in it.

3.2 Chain extrusion $C_p \rightarrow C_{p+1}$

Here we introduce a dimension-independent matrix algorithm for chain extrusion ξ, where *input*: CSR matrix M_p; *output*: CSR matrix M_{p+1}

$$\xi : C_p \rightarrow C_{p+1}; \quad M_p \mapsto M_{p+1}$$

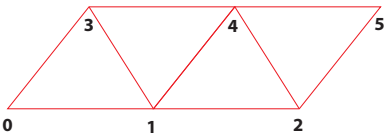


Figure 4: A 2-complex generated by extrusion of the 1-complex $[0,1] \cup [1,2]$.

The extrusion algorithm can be described as follows:

1. *Add a first column of ones* to the input M_p matrix (here $p = 2$):

$$M_2 = \begin{pmatrix} 0 & 1 & 3 \\ 1 & 3 & 4 \\ 1 & 2 & 4 \\ 2 & 4 & 5 \end{pmatrix} \rightarrow \begin{pmatrix} 1 & 0 & 1 & 3 \\ 1 & 1 & 3 & 4 \\ 1 & 1 & 2 & 4 \\ 1 & 2 & 4 & 5 \end{pmatrix} = \widetilde{M}_2.$$

2. *Duplicate the vertex set* V (for a single linear extrusion), by set union with a translated set $\mathbf{T}(\mathbf{t})\, V$[3]:

[3]Otherwise (multiple extrusion) append several properly transformed instances of V. The remaining steps should by updated accordingly.

$$\widehat{V} = V \cup \mathbf{T}(\mathbf{t})\, V.$$

3. Generate a matrix $\widehat{M_p} := \widetilde{M_p}\, E_{r\times 2(p+1)}$ where *rows contain* the indices of the $p+1$ vertices of a *p-cell and of its translated instance*

$$\widehat{M_2} = \widetilde{M_2}\, E = \widetilde{M_2} \begin{pmatrix} 0 & 0 & 0 & 6 & 6 & 6 \\ 1 & 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 \end{pmatrix}.$$

4. *Each row* $\widehat{M}_p^h$ of $\widehat{M_p}$, with $2(p+1)$ elements, *is mapped into* $p+1$ *rows* with $p+2$ elements, by product times a permutation matrix Π^h and a projection matrix P:

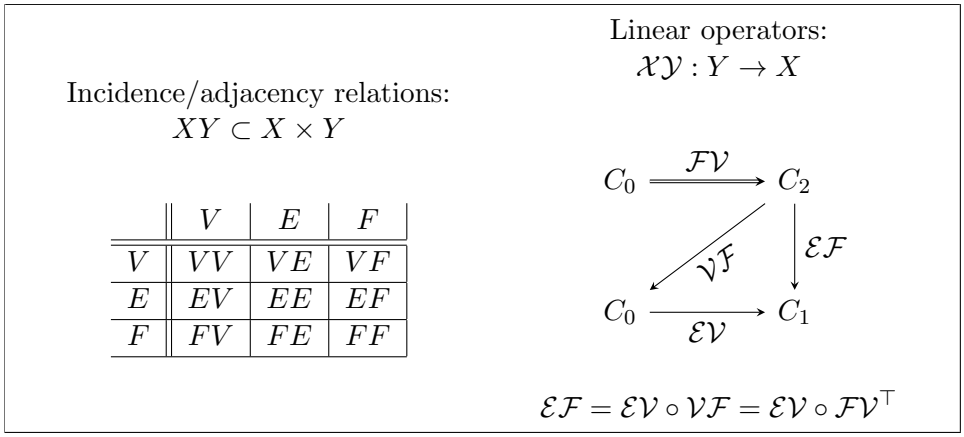
$$M_3 = \xi(M_2) := \bigoplus_h \widehat{M}_2^h\, \Pi^h\, P\ ,$$

where $\Pi^h = \Pi^{h-1}\, \Pi$ and $P : \mathbb{R}^{2(p+1)} \to \mathbb{R}^{p+2}$.

3.2.1 Incidence/adjacency relations

The linear algebraic approach introduced in this paper is a change of paradigm in shape representation: whereas topological relations cannot be easily combined, linear operators are easily composed by product of their matrices. In particular, we show that all incidence and adjacency relations between boundary entities can be computed by simply performing a sparse matrix-matrix multiplication, and that any topological queries — in the sense of [29] — only require a sparse matrix-vector multiplication. It is also well-known that, by using appropriate representations of sparse matrices, such operations can be performed in time linear with the size of the (sparse) output. Furthermore, in recent years, basic algebraic operations between sparse matrices have found hardware support and out-of-the-box fast parallel implementations.

Incidence relations vs linear operators In the remainder of the paper, for the sake of readability, we often use symbols V, E, F for K_0, K_1, K_2, the bases of linear spaces C_0, C_1, C_2 of chains in 2D or in 3D.



	V	E	F
V	VV	VE	VF
E	EV	EE	EF
F	FV	FE	FF

Figure 5: A comparison between binary relations between topological entities V, E, F, and some linear maps between the corresponding chain spaces C_0, C_1, and C_2.

3.3 Incidence/adjacency operators

The simple algebraic manipulation underlying the generation of relational operators between boundary entities is summarised in Figure 5, showing the equivalence between operator composition and matrix product. Notice that only 2-complex structures are displayed here, involving 3 entities V, E, F and 2 fundamental operators $\mathcal{FV}$ and $\mathcal{EV}$.

3.3.1 Dimension-independent $\mathcal{C}_{p,q}$ generation

According to the notations previously used, where we denoted the operators between couple of spaces by using a two-letter identifier to be read backwards (from right to left), let us denote the linear space of p-chains as C_p, that of q-chains as C_q, and the linear operator between them as:

$$\mathcal{C}_{p,q} : C_q \to C_p$$

The whole set of computations needed to generate each incidence or adjacency matrix $[\mathcal{C}_{p,q}]$ between q-chains and p-chains supported by a cellular d-complex is summarised in Table 1. Of course, the actually needed operators depend on the application. The pseudo-code of the algorithm to compute $[\mathcal{C}_{p,q}]$ starting from the input data $[\mathcal{C}_{d,0}]$ and from the query data (p, q) is given in Figure 7. This simple algorithm provides the core functionality of the `topology` attribute of the `model` object in our prototype web

$\mathcal{VV} = \mathcal{VE} \circ \mathcal{EV} = \mathcal{EV}^\top \circ \mathcal{EV}$	$[\mathcal{VV}] = [\mathcal{EV}]^t\,[\mathcal{EV}]$
$\mathcal{VE} = \mathcal{EV}^\top$	$[\mathcal{VE}] = [\mathcal{EV}]^t$
$\mathcal{VF} = \mathcal{FV}^\top$	$[\mathcal{VF}] = [\mathcal{FV}]^t$
$\mathcal{EV}$	$[\mathcal{EV}]$
$\mathcal{EE} = \mathcal{EV} \circ \mathcal{VE} = \mathcal{EV} \circ \mathcal{EV}^\top$	$[\mathcal{EE}] = [\mathcal{EV}]\,[\mathcal{EV}]^t$
$\mathcal{EF} = \mathcal{EV} \circ \mathcal{VF} = \mathcal{EV} \circ \mathcal{FV}^\top$	$[\mathcal{EF}] = [\mathcal{EV}]\,[\mathcal{FV}]^t$
$\mathcal{FV}$	$[\mathcal{FV}]$
$\mathcal{FE} = \mathcal{FV} \circ \mathcal{VE} = \mathcal{FV} \circ \mathcal{EV}^\top$	$[\mathcal{FE}] = [\mathcal{FV}]\,[\mathcal{EV}]^t$
$\mathcal{FF} = \mathcal{FV} \circ \mathcal{VF} = \mathcal{FV} \circ \mathcal{FV}^\top$	$[\mathcal{FF}] = [\mathcal{FV}]\,[\mathcal{FV}]^t$

Figure 6: The derivation of the 9 incidence/adjacency relations between pairs of topological entities in a 2D cell complex (including the boundary representations of 3D solids): (a) the composition of basic operators; (b) the actual computation via products of their matrices.

Table 1: A pictorial representation of the derivation ordering of the various operators between the chain spaces supported by a d-complex. The main information, to be given as input, always concerns the $[\mathcal{C}_{d,0}]$ matrix.

$[\mathcal{C}_{0,0}]$	$\Leftarrow$	$[\mathcal{C}_{1,0}]^t$	$\Leftarrow$	$[\mathcal{C}_{q,0}]^t$	$\Leftarrow$	$[\mathcal{C}_{d,0}]^t$
$\Uparrow$						
$[\mathcal{C}_{1,0}]$				$\Downarrow$		
$\Uparrow$						
$[\mathcal{C}_{p,0}]$		$\Longrightarrow$		$[\mathcal{C}_{p,q}]$		
$\Uparrow$						
$[\mathcal{C}_{d,0}]$						

Data: $[\mathcal{C}_{d,0}],\ (p,q)$
Result: $[\mathcal{C}_{p,q}]$
if $p+q=0$ **then** $k \leftarrow 1$;
else if $p*q=0$ **then** $k \leftarrow p+q$;
else $k \leftarrow \min(p+q)$;
for $h \in [k,d)$ **do**
| $[\mathcal{C}_{k,0}] \leftarrow \phi([\mathcal{C}_{k+1,0}])$;
end
if $p+q=0$ **then return** $[\mathcal{C}_{1,0}]^t[\mathcal{C}_{1,0}]$;
else if $q=0$ **then return** $[\mathcal{C}_{p,0}]$;
else if $p=0$ **then return** $[\mathcal{C}_{q,0}]^t$;
else return $[\mathcal{C}_{p,0}][\mathcal{C}_{q,0}]^t$

Figure 7: Pseudo-code to generate the matrix $[\mathcal{C}_{p,q}]$ $(0 \leq p,q \leq d)$ starting from $[\mathcal{C}_{d,0}]$. In the worst-case the algorithm requires $n-1$ sparse matrix computations and one s.m. product.

implementation of the LAR scheme[4].

3.4 Boundary operator

We distinguish between oriented and unoriented boundary operators ∂_p $(1 \leq p \leq d)$. The first ones are directly computable by sparse matrix-matrix multiplication of the $[C_{p-1,0}]$ and $[C_{p,0}]^t$ matrices. The second one requires to compute the orientation of each p-cell. In case of d-simplices in $\mathbb{E}^d$ this operation requires a $(d+1) \times (d+1)$ matrix inversion for each d-cell.

3.4.1 Unoriented boundary

It is defined for chains over the commutative group $\mathbb{Z}_2 = \{0,1\}$, i.e. *without cell orientation*. In the remainder we use the symbol $\mathbb{Z}_2$ as a function: $\mathbb{Z}_2 : \mathbb{Z} \to \{0,1\};\ k \mapsto (k \bmod 2)$.

1. As a first step compute $[\mathcal{C}_{p-1,p}] := [C_{p-1,0}][C_{p,0}]^t$, and then set, for $1 \leq i \leq k_{p-1}$, and $1 \leq j \leq k_p$:

$$[\widetilde{\partial}_p](i,j) = \begin{cases} 1 & \text{if} \quad [\mathcal{C}_{p-1,p}](i,j) = \max_h\ [\mathcal{C}_{p-1,p}](i,h) \\ 0 & \text{else} \end{cases}$$

[4] See https://github.com/cvdlab/lar and/or https://github.com/cvdlab/lar-demo.

2. Then compose $\widetilde{\partial}_p$ with the "Boolean transformation" $\mathbb{Z}_2$

Definition 11 (Unoriented boundary operator)**.** *Boundary operator "without orientation" is*

$$\partial_p := \mathbb{Z}_2 \circ \widetilde{\partial}_p$$

3.4.2 Oriented boundary

It is defined for chains over the commutative group $\mathbb{Z}$ of (implicitly) oriented cells.

1. Let every cell σ_p in a p-complex K embedded in $\mathbb{R}^p$ have the *intrinsic orientation* defined by its *canonical* representation:

$$\sigma_p^h =< \sigma_0^{h_0}, \sigma_0^{h_1}, \ldots, \sigma_0^{h_p} > \qquad \text{with} \qquad h_0 < h_1 < \cdots, < h_p$$

2. Compute the absolute orientation χ of every p-simplex σ_p^h, by using the matrices V_p^h of *homogeneous coordinates* of its vertices:

$$\chi : h_p \to \{-1, 1\}; \quad \sigma_p^h \mapsto (\text{sign} \circ \det)(V_p^h)$$

3. Finally, consider the unoriented boundary matrix $[\partial_p] : \mathcal{C}_p \to \mathcal{C}_{p-1}$, and assign to each *unit term in position* i, h the value

$$a_{i,h} := \rho(\sigma_{p-1}^i, \sigma_p^h) = (-1)^{\ell} \cdot \chi(\sigma_p^h) \in \{-1, 1\}, \qquad \text{with} \qquad \ell = \sum_{m<h} a_{i,m}$$

where the function $\rho : K_{p-1} \times K_p \to \{-1, 1\}$ provides the *relative orientation* of the cell σ_{p-1}^i with respect to the cell σ_p^h, and the index ℓ is computed by summing over the terms of the *binary* boundary matrix $[\partial_p]$.

4 Examples

In this section we discuss few elementary examples, in order to give some cue about the simple computations required to get full knowledge about the topology of geometric objects represented as cellular complexes.

4.1 2D simplicial complex (solid & non-manifold pointset)

First consider a 2D pointset, defined as a simplicial complex that is locally non manifold, i.e. with some points that have neighborouds that are non homeomorphic to the 2-ball. Let K_0, K_1, K_2 be the sets of 0-, 1-, and 2-cells, respectively, with $K = K_0 \cup K_1 \cup K_2$ and with $\#K_0 =: k_0 = 9$, $\#K_1 =: k_1 = 16$, and $\#K_2 =: k_2 = 6$.

For a simplicial complex, the whole object topology can be computed giving as input only the CSR matrix representation of the chain corresponding to the whole K_2 set, that we have characterised as the $[C_{2,0}]$ matrix. Both the input dataset `FV` and the derived `EV` relation are given in Figure 8, together with an image of the pointset with numbered topological elements.

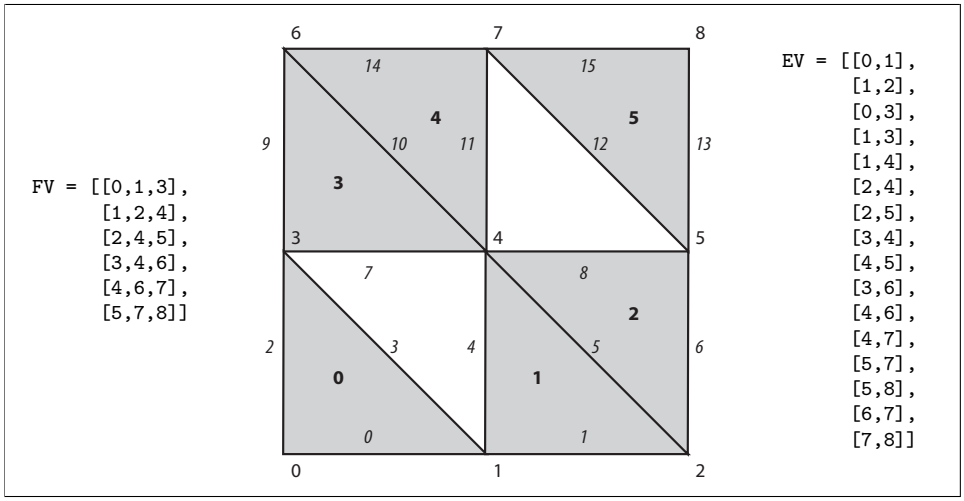


Figure 8: 2-complex example: (a) CSR representation of $[C_{2,0}]$ — describing the 2-cells (better: the 2-chain space generated by the matrix rows) via its 0-cells (better: the 0-chain space generated by the matrix columns). Notice that CSR($C_{2,0}$) — i.e. `FV` — is the only needed input; (b) drawing of the complex, with numbered 0-, 1-, and 2-cells; (c) the 1-cells via the 0-cells, i.e. the CSR representation of $[C_{1,0}]$ computed as $\phi(C_{2,0})$.

Characteristic matrices An explicit description of the sparse binary matrices $[\mathcal{EV}]$ and $[\mathcal{FV}]$ of dimensions $k_1 \times k_0$ and $k_2 \times k_0$ is given below. They can be characterised as the coordinate representation, with respect to the standard p-chain basis — i.e. the single p-cells ($0 \le p \le 2$) — of the operators $\mathcal{EV} : C_0 \to C_1$ and $\mathcal{FV} : C_0 \to C_2$ between spaces of p-chains.

$$[\mathcal{EV}]=\begin{pmatrix} 1&1&0&0&0&0&0&0&0\\ 0&1&1&0&0&0&0&0&0\\ 1&0&0&1&0&0&0&0&0\\ 0&1&0&1&0&0&0&0&0\\ \vdots&\vdots&\vdots&\vdots&\vdots&\vdots&\vdots&\vdots&\vdots\\ 0&0&0&0&0&1&0&1&0\\ 0&0&0&0&0&1&0&0&1\\ 0&0&0&0&0&0&1&1&0\\ 0&0&0&0&0&0&0&1&1 \end{pmatrix} \qquad [\mathcal{FV}]=\begin{pmatrix} 1&1&0&1&0&0&0&0&0\\ 0&1&1&0&1&0&0&0&0\\ 0&0&1&0&1&1&0&0&0\\ 0&0&0&1&1&0&1&0&0\\ 0&0&0&0&1&0&1&1&0\\ 0&0&0&0&0&1&0&1&1 \end{pmatrix}$$

```
VV =
[[0,1,3],
 [0,1,2,3,4],
 [1,2,4,5],
 [0,1,3,4,6],
 [1,2,3,4,5,6,7],
 [2,4,5,7,8],
 [3,4,6,7],
 [4,5,6,7,8],
 [5,7,8]]
```

```
EV =
[[0,2],
 [0,1,3,4],
 [1,5,6],
 [2,3,7,9],
 [4,5,7,8,10,11],
 [6,8,12,13],
 [9,10,14],
 [11,12,14,15],
 [13,15]]
```

```
FV =
[[0],
 [0,1],
 [1,2],
 [0,3],
 [1,2,3,4],
 [2,5],
 [3,4],
 [4,5],
 [5]]
```

```
VE =
[[0,1],
 [1,2],
 [0,3],
 [1,3],
 [1,4],
 [2,4],
 [2,5],
 [3,4],
 [4,5],
 [3,6],
 [4,6],
 [4,7],
 [5,7],
 [5,8],
 [6,7],
 [7,8]]
```

```
EE =
[[0,1,2,3,4],
 [0,1,3,4,5,6],
 [0,2,3,7,9],
 [0,1,2,3,4,7,9],
 [0,1,3,4,5,7,8,10,11],
 [1,4,5,6,7,8,10,11],
 [1,5,6,8,12,13],
 [2,3,4,5,7,8,9,10,11],
 [4,5,6,7,8,10,11,12,13],
 [2,3,7,9,10,14],
 [4,5,7,8,9,10,11,14],
 [4,5,7,8,10,11,12,14,15],
 [6,8,11,12,13,14,15],
 [6,8,12,13,15],
 [9,10,11,12,14,15],
 [11,12,13,14,15]]
```

```
FE =
[[0,1],
 [0,1,2],
 [0,3],
 [0,1,3],
 [0,1,2,3,4],
 [1,2,3,4],
 [1,2,5],
 [0,1,2,3,4],
 [1,2,3,4,5],
 [0,3,4],
 [1,2,3,4],
 [1,2,3,4,5],
 [2,4,5],
 [2,5],
 [3,4,5],
 [4,5]]
```

```
VF =
[[0,1,3],
 [1,2,4],
 [2,4,5],
 [3,4,6],
 [4,6,7],
 [5,7,8]]
```

```
EF =
[[0,1,2,3,4,7,9],
 [0,1,3,4,5,6,7,8,10,11],
 [1,4,5,6,7,8,10,11,12,13],
 [2,3,4,5,7,8,9,10,11,14],
 [4,5,7,8,9,10,11,12,14,15],
 [6,8,11,12,13,14,15]]
```

```
FF =
[[0,1,3],
 [0,1,2,3,4],
 [1,2,3,4,5],
 [0,1,2,3,4],
 [1,2,3,4,5],
 [2,4,5]]
```

Figure 9: The CSR representation of the operators $\mathcal{C}_{p,q}$, with $0 \leq p \leq 2$ and $0 \leq q \leq 2$, for the the model given in Figure 8. Notice that they directly correspond to the 3×3 binary relations between the topological entities V, E, F.

Incidence on vertices The three operators corresponding to relations `VV`, `VE`, and `VF` for the simplicial compilex of Figure 8 are given in CSR matrix form in the top partition of Figure 9. Their signature is:

$$\mathcal{VV} : C_0 \to C_0, \qquad \mathcal{EV} : C_0 \to C_1, \qquad \mathcal{FV} : C_0 \to C_2$$

It is mandatory to remark that Figure 9 provides only a partial picture of the CSR matrix representation, showing for each row the column indices of the non-zero elements, but giving no value, that is often not equal to one.

Incidence on edges Very similar sparse matrix representations (see Figure 9, middle partition) are obtained for the operators that associate the incident vertices, edges and faces (better: the incident 0-, 1-, and 2-chains) to every 1-chain, i.e. for:

$$\mathcal{VE} : C_1 \to C_0, \qquad \mathcal{EE} : C_1 \to C_1, \qquad \mathcal{FE} : C_1 \to C_2$$

Incidence on faces and analogously (see Figure 9, bottom partition) for the operators that associate the incident 0-, 1-, and 2-chains to each 2-chain:

$$\mathcal{VF} : C_2 \to C_0, \qquad \mathcal{EF} : C_2 \to C_1, \qquad \mathcal{FF} : C_2 \to C_2$$

Examples of topological queries To perform a topological query requires a matrix-vector multiplication between a sparse matrix and a sparse column vector:

$$[c_0^k] = [0, \ldots, 1, \ldots, 0]^t \qquad \text{(0-chain basis element)}$$

$$\mathcal{VV}(c_0^k) \equiv [\mathcal{VV}][c_0^k]; \qquad \mathcal{EV}(c_0^k) \equiv [\mathcal{EV}][c_0^k]; \qquad \mathcal{FV}(c_0^k) \equiv [\mathcal{FV}][c_0^k])$$

$$[c_1^k] = [0, \ldots, 1, \ldots, 0]^t \qquad \text{(1-chain basis element)}$$

$$\mathcal{VE}(c_1^k) = [\mathcal{VE}][c_1^k]; \qquad \mathcal{EE}(c_1^k) = [\mathcal{EE}][c_1^k]; \qquad \mathcal{FE}(c_1^k) = [\mathcal{FE}][c_1^k]$$

$$[c_2^k] = [0, \ldots, 1, \ldots, 0]^t \qquad \text{(2-chain basis element)}$$

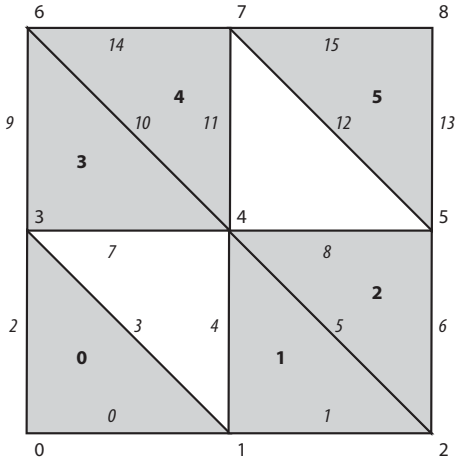
$$\mathcal{VF}(c_2^k) = [\mathcal{VF}][c_2^k]; \qquad \mathcal{EF}(c_2^k) = [\mathcal{EF}][c_2^k]; \qquad \mathcal{FF}(c_2^k) = [\mathcal{FF}][c_2^k]$$

$[\mathcal{EF}] =$

```
[[2 1 0 0 0 0]
 [1 2 1 0 0 0]
 [2 0 0 1 0 0]
 [2 1 0 1 0 0]
 [1 2 1 1 1 0]
 [0 2 2 1 1 0]
 [0 1 2 0 0 1]
 [1 1 1 2 1 0]
 [0 1 2 1 1 1]
 [1 0 0 2 1 0]
 [0 1 1 2 2 0]
 [0 1 1 1 2 1]
 [0 0 1 0 1 2]
 [0 0 1 0 0 2]
 [0 0 0 1 2 1]
 [0 0 0 0 1 2]]
```

$[\partial_2] =$

```
[[1 0 0 0 0 0]
 [0 1 0 0 0 0]
 [1 0 0 0 0 0]
 [1 0 0 0 0 0]
 [0 1 0 0 0 0]
 [0 1 1 0 0 0]
 [0 0 1 0 0 0]
 [0 0 0 1 0 0]
 [0 0 1 0 0 0]
 [0 0 0 1 0 0]
 [0 0 0 1 1 0]
 [0 0 0 0 1 0]
 [0 0 0 0 0 1]
 [0 0 0 0 0 1]
 [0 0 0 0 1 0]
 [0 0 0 0 0 1]]
```



$[\mathcal{VE}] =$

```
[[1 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]
 [1 1 0 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0]
 [0 1 0 0 0 1 1 0 0 0 0 0 0 0 0 0 0 0]
 [0 0 1 1 0 0 0 1 0 1 0 0 0 0 0 0 0 0]
 [0 0 0 0 1 1 0 1 1 0 1 1 0 0 0 0 0 0]
 [0 0 0 0 0 0 1 0 1 0 0 0 1 1 0 0 0 0]
 [0 0 0 0 0 0 0 0 0 1 1 0 0 0 1 0 0 0]
 [0 0 0 0 0 0 0 0 0 0 0 1 1 0 1 1 0 0]
 [0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 1 1 1]]
```

$[\partial_1] =$

```
[[1 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]
 [1 1 0 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0]
 [0 1 0 0 0 1 1 0 0 0 0 0 0 0 0 0 0 0]
 [0 0 1 1 0 0 0 1 0 1 0 0 0 0 0 0 0 0]
 [0 0 0 0 1 1 0 1 1 0 1 1 0 0 0 0 0 0]
 [0 0 0 0 0 0 1 0 1 0 0 0 1 1 0 0 0 0]
 [0 0 0 0 0 0 0 0 0 1 1 0 0 0 1 0 0 0]
 [0 0 0 0 0 0 0 0 0 0 0 1 1 0 1 1 0 0]
 [0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 1 1 1]]
```

Figure 10: The matrix representations of the ∂_2 and ∂_1 operators for the simplicial complex displayed in this figure. Notice that $[\partial_1]$ is equal to $[\mathcal{VE}]$, since the max value of each row is 1.

Boundary operators Let us remember that

$$\partial_p : C_p \to C_{p-1}, \qquad 1 \leq p \leq n$$

are linear operators to compute the $(p-1)$-cycle that is the boundary of a n-chain. Also remember that the algorithm given in Section 3.4.1 provides a matrix representation of the *unoriented* boundary operator.

Some examples of boundary computation are shown in Figure 11: take the 2-chains $c_2, d_2, e_2 \in C_2$, defined in coordinates as:

$$\begin{aligned} [c_2] &= [1,1,1,1,1,1]^t \\ [d_2] &= [1,0,0,0,0,0]^t \\ [e_2] &= [0,0,0,1,1,1]^t \end{aligned} \tag{2}$$

and compute, according to Section 3.4.1, the expressions:

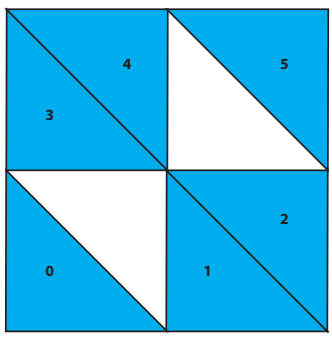
$$\begin{aligned} [c_1] &= \mathbb{Z}_2\left([\partial_2][c_2]^t\right) = [1,1,1,1,1,0,1,1,1,1,0,1,1,1,1,1]^t \\ [d_1] &= \mathbb{Z}_2\left([\partial_2][d_2]^t\right) = [1,0,1,1,0,0,0,0,0,0,0,0,0,0,0,0]^t \\ [e_1] &= \mathbb{Z}_2\left([\partial_2][e_2]^t\right) = [0,0,0,0,0,0,0,1,0,1,0,1,1,1,1,1]^t \end{aligned} \tag{3}$$

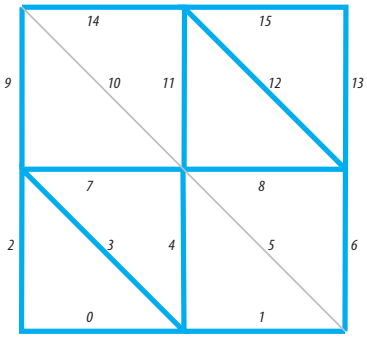
The input 2-chains (3) and the geometric results (4) of the computations are shown in Figures 11a, 11b, and 11c, respectively.

The boundary of a boundary is empty ($\partial\partial = \mathbf{0}$) Of course, the constraint equations of chain complexes, requiring that the image of ∂_p is contained in the kernel of ∂_{p-1} $(1 \leq p \leq n)$, are satisfied. Here we have:

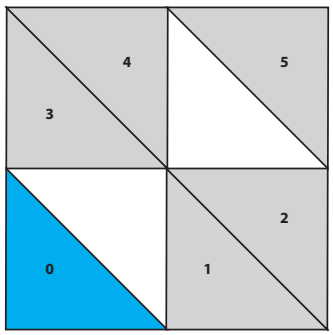
$$[\partial_1]\,[\partial_2]\,\left[\begin{array}{ccc} c_2 & d_2 & e_2 \end{array}\right] =$$

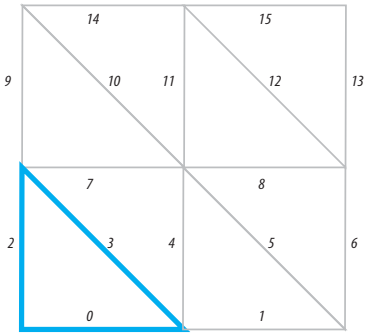
$$= \begin{array}{c}\texttt{[[1 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0]}\\ \texttt{[1 1 0 1 1 0 0 0 0 0 0 0 0 0 0 0]}\\ \texttt{[0 1 0 0 0 1 1 0 0 0 0 0 0 0 0 0]}\\ \texttt{[0 0 1 1 0 0 0 1 0 1 0 0 0 0 0 0]}\\ \texttt{[0 0 0 0 1 1 0 1 1 0 1 1 0 0 0 0]}\\ \texttt{[0 0 0 0 0 0 1 0 1 0 0 0 1 1 0 0]}\\ \texttt{[0 0 0 0 0 0 0 0 0 1 1 0 0 0 1 0]}\\ \texttt{[0 0 0 0 0 0 0 0 0 0 0 1 1 0 1 1]}\\ \texttt{[0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 1]]}\end{array} \times \begin{array}{c}\texttt{[[1 0 0 0 0 0]}\\ \texttt{[0 1 0 0 0 0]}\\ \texttt{[1 0 0 0 0 0]}\\ \texttt{[1 0 0 0 0 0]}\\ \texttt{[0 1 0 0 0 0]}\\ \texttt{[0 1 1 0 0 0]}\\ \texttt{[0 0 1 0 0 0]}\\ \texttt{[0 0 0 1 0 0]}\\ \texttt{[0 0 1 0 0 0]}\\ \texttt{[0 0 0 1 0 0]}\\ \texttt{[0 0 0 1 1 0]}\\ \texttt{[0 0 0 0 1 0]}\\ \texttt{[0 0 0 0 0 1]}\\ \texttt{[0 0 0 0 0 1]}\\ \texttt{[0 0 0 0 1 0]}\\ \texttt{[0 0 0 0 0 1]]}\end{array} \times \begin{array}{c}\texttt{[[1 1 0]}\\ \texttt{[1 1 0]}\\ \texttt{[1 0 0]}\\ \texttt{[1 0 1]}\\ \texttt{[1 0 1]}\\ \texttt{[1 0 1]]}\end{array} = \begin{array}{c}\texttt{[[2 2 0]}\\ \texttt{[4 2 0]}\\ \texttt{[4 0 0]}\\ \texttt{[4 2 2]}\\ \texttt{[8 0 4]}\\ \texttt{[4 0 2]}\\ \texttt{[4 0 4]}\\ \texttt{[4 0 4]}\\ \texttt{[2 0 2]]}\end{array}$$



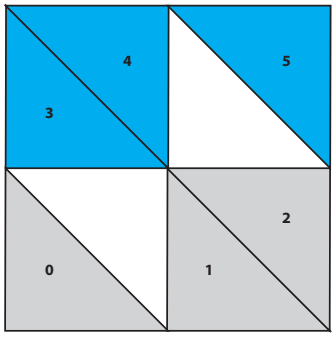


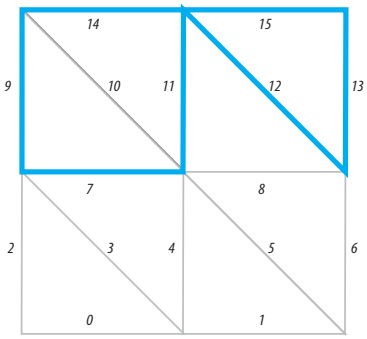
$$[c_1] = \mathbb{Z}_2\left([\partial_2][1,1,1,1,1,1]^t\right) = [1,1,1,1,1,0,1,1,1,1,0,1,1,1,1,1]^t$$





$$[d_1] = \mathbb{Z}_2\left([\partial_2][1,0,0,0,0,0]^t\right) = [1,0,1,1,0,0,0,0,0,0,0,0,0,0,0,0]^t$$





$$[e_1] = \mathbb{Z}_2\left([\partial_2][0,0,0,1,1,1]^t\right) = [0,0,0,0,0,0,0,1,0,1,0,1,1,1,1,1]^t$$

Figure 11: The computation of boundary chains c_1, d_1, e_1, starting from 2-chains c_2, d_2, e_2, respectively.

and, by applying the "Booleanizer" operator $\mathbb{Z}_2 : \mathbb{Z} \to \{0, 1\}$, we finally get:

$$\mathbb{Z}_2([\partial_1]\,[\partial_2]\,\begin{bmatrix} c_2 & d_2 & e_2 \end{bmatrix}) = \mathbb{Z}_2(\begin{bmatrix} 2 & 2 & 0 \\ 4 & 2 & 0 \\ 4 & 0 & 0 \\ 4 & 2 & 2 \\ 8 & 0 & 4 \\ 4 & 0 & 2 \\ 4 & 0 & 4 \\ 4 & 0 & 4 \\ 2 & 0 & 2 \end{bmatrix}) = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}$$

4.2 The "house-with-two-rooms"

An interesting example of two-dimensional CW-complex embedded in $\mathbb{E}^3$ is given by Hatcher in Reference [11]. The book shows that such 2-complex $K = K_0 \cup K_1 \cup K_2$, with 29 0-cells, 51 1-cells, and 23 2-cells, is contractible to a point. Of course, the "house-with-two-rooms" example is both non-solid and non-manifold.

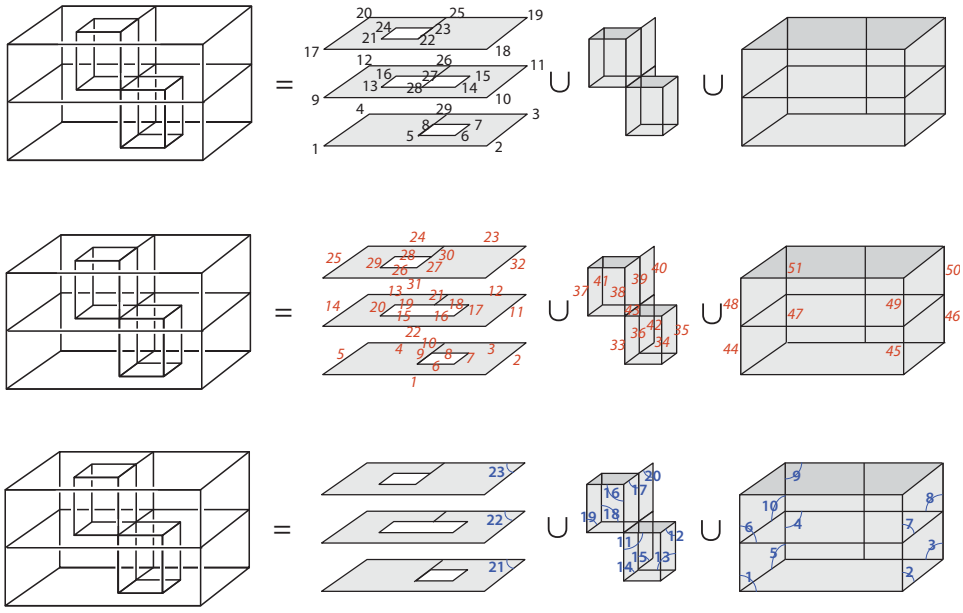


Figure 12: The numeric coding of the 0-, 1-, and 2-cells of the "house-with-two-rooms" example, respectively.

The input representation In this example the `EV` dataset is not (at least, easily) computable from the CSR representation `FV` of the $C_{2,0}$ matrix, since

the 2-cells are not all convex. Therefore, we give it as explicit input, so that the topology is represented in this case by the pair (`EV`, `FV`).

```
EV = [[1,2],[2,3],[3,29],[4,29],[1,4],[5,6],[6,7],[7,8],[5,8],[8,29
],[10,11],[11,26],[12,26],[9,12],[13,28],[28,14],[14,15],[15,27],
[16,27],[13,16],[26,27],[9,10],[19,25],[20,25],[17,20],[21,22],[
22,23],[23,24],[21,24],[23,25],[17,18],[18,19],[5,28],[6,14],[7,
15],[8,27],[13,21],[22,28],[23,27],[25,26],[16,24],[26,29],[27,28
],[1,9],[2,10],[3,11],[4,12],[9,17],[10,18],[11,19],[12,20]]
```

```
FV = [[1,2,9,10],[2,3,10,11],[13,11,26,29],[4,12,26,29],[1,4,9,12],
[9,10,17,18],[10,11,18,19],[11,19,25,26],[12,20,25,26],[9,12,17,
20],[5,6,14,28],[6,7,14,15],[7,8,15,27],[5,8,27,28],[8,26,27,29],
[13,21,22,28],[22,23,27,28],[16,23,24,27],[13,16,21,24],[23,25,26
,27],[1,2,3,4,5,6,7,8,29],[9,10,11,12,13,14,15,16,26,27,28],[17,
18,19,20,21,22,23,24,25]]
```

Conversely, since the geometric embedding is piecewise affine, the only geometric representation required for the cells consists in the $\mathbb{E}^3$ position of vertices, i.e. of 0-cells of the complex:

```
V = [[3.0,0.0,0.0],[3.0,4.0,0.0],[0.0,4.0,0.0],[0.0,0.0,0.0],[2.0,2
.0,0.0],[2.0,3.0,0.0],[1.0,3.0,0.0],[1.0,2.0,0.0],[3.0,0.0,1.0],[
3.0,4.0,1.0],[0.0,4.0,1.0],[0.0,0.0,1.0],[2.0,1.0,1.0],[2.0,3.0,1
.0],[1.0,3.0,1.0],[1.0,1.0,1.0],[3.0,0.0,2.0],[3.0,4.0,2.0],[0.0,
4.0,2.0],[0.0,0.0,2.0],[2.0,1.0,2.0],[2.0,2.0,2.0],[1.0,2.0,2.0],
[1.0,1.0,2.0],[0.0,2.0,2.0],[0.0,2.0,1.0],[1.0,2.0,1.0],[2.0,2.0,
1.0],[0.0,2.0,0.0]]
```

Boundary computation of several chains The boundaries of several 2-chains are computed and displayed in Figure 13, starting from chains of smaller size, and ending with the 2-chain corresponding to the topological sum of all the 2-cells. It is worth noting that the boundary of such last chain suggests a simple *homotopy retraction* to reduce to a point the whole space, that was certainly not evident looking directly at this space.

4.3 Extrusion and boundary extraction

Here we discuss some simple examples of geometric computations using the representation scheme based on chain complexes introduced in this paper.

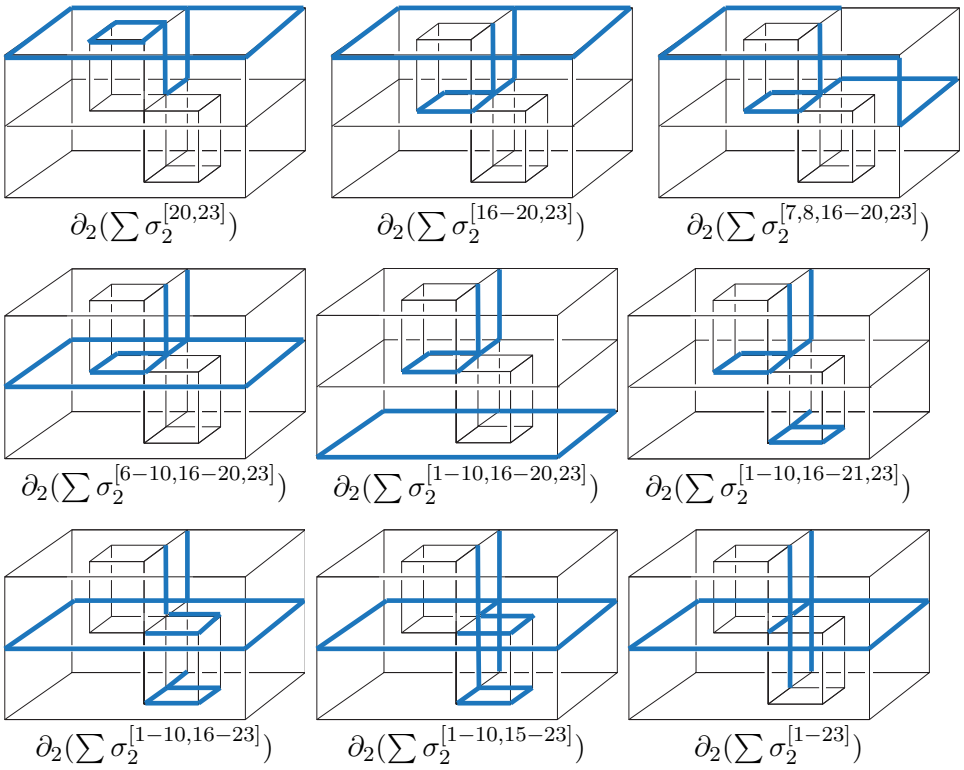


Figure 13: A graphical representation of the 1-chains computed by mapping, via the boundary operator ∂_2, different 2-chains of the well-known topological example. The last one (at bottom right corner) is the boundary chain associated to the whole surface. As it is possible to see, it shows a simple way to perform the reduction of the surface to a point via homotopic retraction.

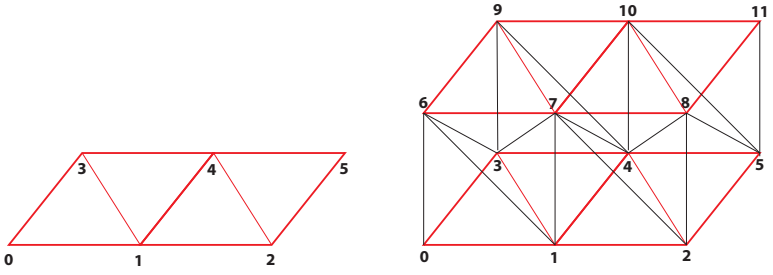


Figure 14: The 3-complex generated by extrusion of the 2-complex on the left side of the picture.

4.3.1 Structured meshes

Let call

$$\xi : C_p \to C_{p+1}$$

the (dimension-independent) extrusion operator from p-chains to $(p+1)$-chains introduced in Section 3.2. Here we apply twice this operator, starting from a 1-complex and producing the generation of a simplicial 3-complex. In particular, we start from the CSR representation of a chain c_1, where

$$c_1 = \begin{pmatrix} 0 & 1 \\ 1 & 2 \end{pmatrix}$$

Hence we get a 2-chain c_2, shown in Figure 14a, as:

$$c_2 = \xi(c_1) = \begin{pmatrix} 0 & 1 & 3 \\ 1 & 2 & 4 \\ 1 & 3 & 4 \\ 2 & 4 & 5 \end{pmatrix}$$

A second extrusion operation generates a 3-chain. Notice that the input c_2 chain has 4 two-cells, so that the resulting $\xi(c_2)$ chain contains $3 \times 4 = 12$ three-cells well glued together, as shown in Figure 14b:

$$c_3 = \xi(c_2) = \begin{pmatrix} 0 & 1 & 3 & 6 \\ 1 & 2 & 4 & 7 \\ 1 & 3 & 4 & 7 \\ 1 & 3 & 6 & 7 \\ 2 & 4 & 5 & 8 \\ 2 & 4 & 7 & 8 \\ 3 & 4 & 7 & 9 \\ 3 & 6 & 7 & 9 \\ 4 & 5 & 8 & 10 \\ 4 & 7 & 8 & 10 \\ 4 & 7 & 9 & 10 \\ 5 & 8 & 10 & 11 \end{pmatrix}$$

This kind of cell complexes are normally used for regular domain decompositions in parametric geometry. It is worth noting that it is very simple to generate 4D or even higher-dimensional cell complexes using the ξ operator.

4.3.2 Unstructured meshes

Some examples of handling of unstructured meshes is given in this section. In the 2D example shown in Figure 15 a triangulation of a set of points, introduced as a CSR matrix representation `FV` is described in Figure 15a, then the 1-boundary chain is extracted in Figure 15b, whereas a simplicial complex triangulating the 3-chain $\xi($`FV`$)$ is shown in Figure 15c.

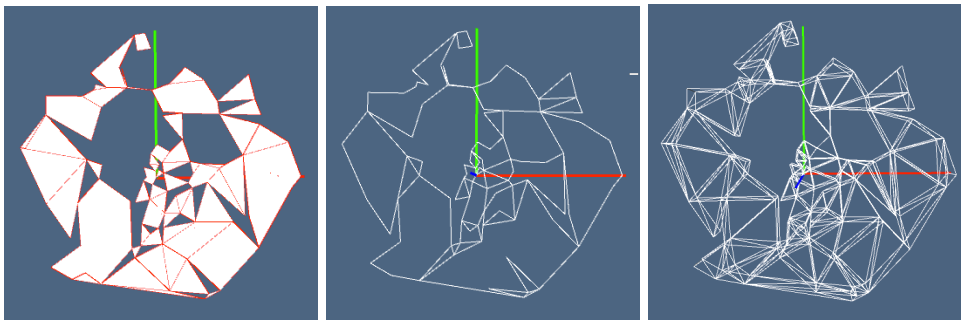


Figure 15: (a) A triangulated polygon; (b) the boundary 1-complex; (c) the 1-skeleton of the extruded triangulation.

4.3.3 Random polygon

It is more interesting to generate a simple 2D polygon of very high Euler characteristic $\chi = k_0 - k_1 + k_2$. In order to produce such dataset to test our representation scheme, we start by generating a set `V` of 10^4 random points within the open 2-disk

$$D^2 = \{(x, y) : x^2 + y^2 < 1\},$$

then we produce a Delaunay triangulation of `V`, finally removing a random subset of triangles from the generated triangulation. The resulting solid and non-manifold polygon $[C_{2,0}]$, described by the CSR representation as a 2-chain:

c_2 = `[[5797, 4303, 6805], [7597, 5864, 3760], ... , [3036, 1795, 8392]],`

is shown in Figure 16a. The unoriented boundary chain $c_1 = \mathbb{Z}_2(\partial(c_2))$ is shown in Figure 16b. The *oriented boundary 2-chain* $b_3 = \partial(\xi(c_2))$ and an unoriented 2-chain $d_2 = \xi(c_1)$ are shown in Figures 16c and 16d, respectively. The reader should notice that the Euler number (or Euler characteristic) of such chains is quite interesting.

4.3.4 Random polyhedron

A set of p-simplices is said *coherently oriented* when their common faces have opposite orientation. In this example, a chain of 3-simplices is randomly produced in $\mathbb{E}^3$, where a set of 10^5 random points is first generated, then the Delaunay tetrahedra produced by such points are filtered against the

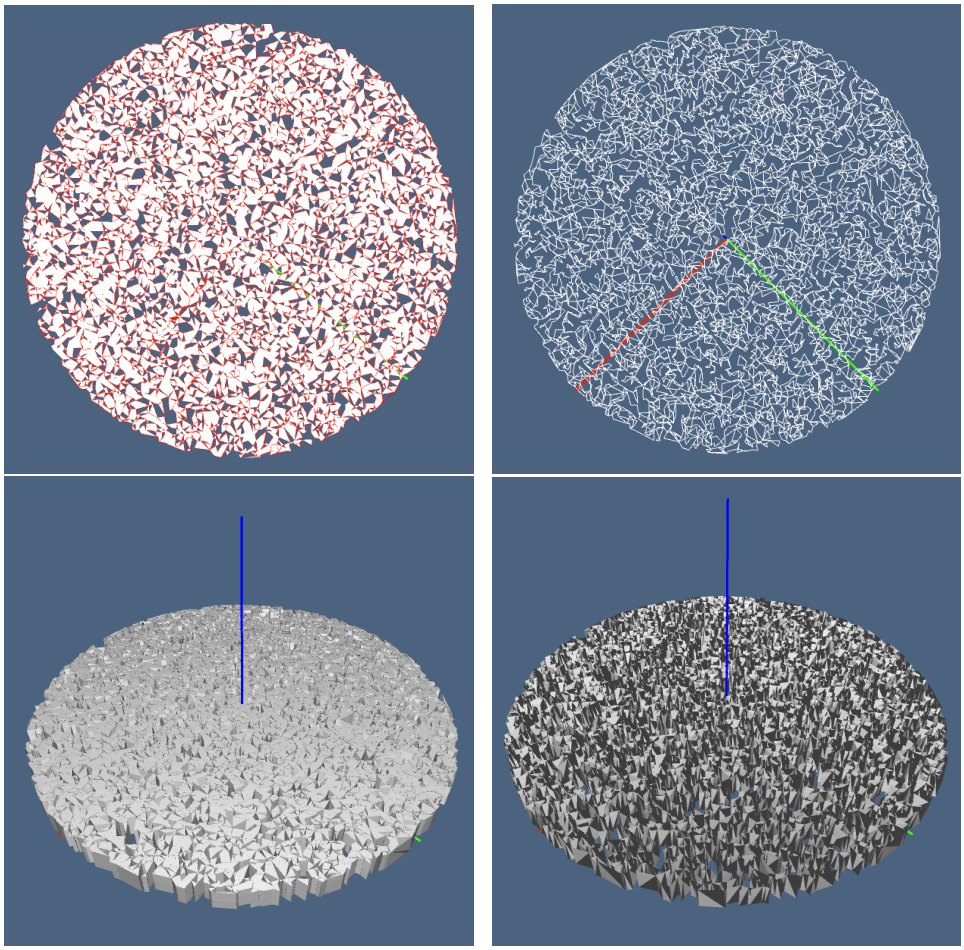


Figure 16: (a) A random polygon c_2; (b) its boundary $c_1 = (\mathbb{Z}_2 \circ \partial)(c_2)$; (c) the extruded polyhedron $b_3 = \xi(c_2)$; (d) the extruded boundary $d_2 = \xi(c_1)$.

subspace $z \geq 0$, so that the only ones fully contained in the subspace $z < 0$ are retained in the non-coherently oriented output chain, that is named c_3 and is displayed in Figure 17a. To recover a *coherent orientation* of the boundary cells $c_2 = (\mathbb{Z}_2 \circ \partial)(c_3)$ is quite easy. Just consider the (unique) coboundary 3-cell of boundary 2-cells, and test its orientation, by computing the sign of the determinant of the matrix of the homogeneous representation of vertices. Then possibly invert the orientation of the face depending on the permutation class (even or odd) of its missing vertex, according to the method discussed in Section 3.4.2.

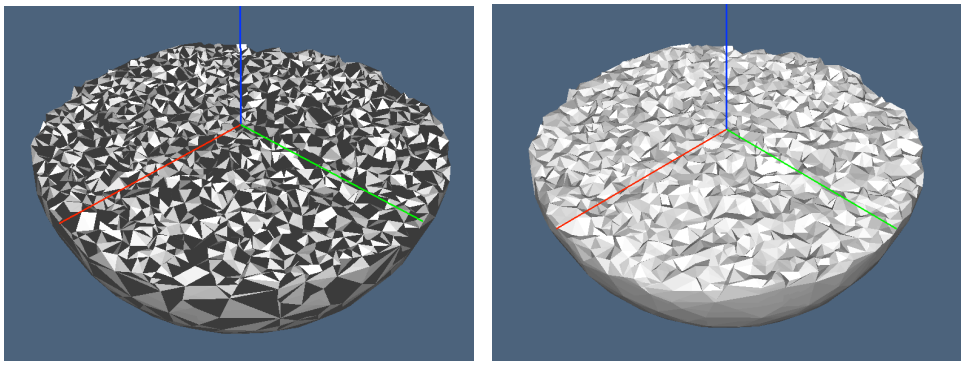


Figure 17: (a) A random 3-complex, with no coherent orientation of 3-simplices, within the lower half of the D^3 disk; (b) its (coherently) *oriented* 2-boundary surface.

5 Conclusion

In this paper we have introduced a simple algebraic representation scheme for CW-complexes, represented using a CSR matrix (Compressed Sparse Row) form of the characteristic matrices of the chain of higher-dimensional cells and/or of chains of their lower-dimensional faces. Such algebraic representation allows to either compute globally or query locally the topology of the complex using very simple algebraic tools, mainly based on the sparse matrix-matrix or matrix-vector multiplication. This approach has several strengths. First and more important, it may support not only boundary representations, but also cellular decompositions of the interior. Then, whereas we have restricted here our consideration mainly to simplicial complexes, much

wider domains can be represented with sparse binary matrices, and appropriate characteristic maps, so that it can handle much general kinds of piecewise-linear cells, and is extensible to curved cells. Also, our approach is dimension-independent, and not restricted only to regular cellular complexes but allows internal structures, including cracks, missing points and fibres of different dimensions, and is extensible to curved cells using standard parametric tools, i.e. polynomial and rational maps. From a technological viewpoint, it can employ off-the-shelf hardware/software support for parallel implementation and fast graphics, including OpenCL, OpenGL, and their web counterparts, because of the absolute lack of need of algorithms for traversing or searching linked data structures. Last but not least, our algebraic representation of geometric structures enjoys a closed math form, using *chains* — the geometric support of discrete integrals — and being extensible to *cochains* —- the discrete representation of integrand differential forms — and can therefore be used for joint representation of geometric and physical properties, opening the door to possible further advances and breakthroughs.

Acknowledgements

We would like to acknowledge our collaborators Enrico Marino and Federico Spini for all their support, in particular for developing a very first prototype API for chain-based geometric programming in Javascript, and a very simple web application for testing the approach without the need to install any plugin or software library. The API can be downloaded from https://github.com/cvdlab/lar; the demo web application, is accessible from https://github.com/cvdlab/lar-demo.

References

[1] Ala, S. R. Performance anomalies in boundary data structures. *IEEE Comput. Graph. Appl. 12*, 2 (Mar. 1992), 49–58.

[2] Bajaj, C., DiCarlo, A., Haiat, G., Laugier, P., Milicchio, F., Naili, S., Padilla, F., Paoluzzi, A., Peyrin, F., and Scorzelli, G. Extracting trabecular geometry from to-

mographic images of spongy bone. *Journal of Biomechanics 39, Supplement 1* (2006).

[3] Baumgart, B. G. Winged edge polyhedron representation. Tech. Rep. Stan-CS-320, Stanford, CA, USA, 1972.

[4] Bowyer, A., and Geometric Modelling Society. *Introducing Djinn: A Geometric Interface for Solid Modelling*. Information Geometers [for] the Geometric Modelling Society, 1995.

[5] Braid, I. C. The synthesis of solids bounded by many faces. *Commun. ACM 18*, 4 (Apr. 1975), 209–216.

[6] Desbarats, P., and Gueorguieva, S. CW complexes: topological mainframe for numerical representations of objects. In *Proceedings of the 2003 international conference on Computational science and its applications: PartIII* (Berlin, Heidelberg, 2003), ICCSA'03, Springer-Verlag, pp. 498–507.

[7] DiCarlo, A., Milicchio, F., Paoluzzi, A., and Shapiro, V. Chain-based representations for solid and physical modeling. *Automation Science and Engineering, IEEE Transactions on 6*, 3 (July 2009), 454 –467.

[8] Dobkin, D. P., and Laszlo, M. J. Primitives for the manipulation of three-dimensional subdivisions. In *Proceedings of the third annual symposium on Computational geometry* (New York, NY, USA, 1987), SCG '87, ACM, pp. 86–99.

[9] Gomes, A., Middleditch, A., and Reade, C. A mathematical model for boundary representations of n-dimensional geometric objects. In *Proceedings of the fifth ACM symposium on Solid modeling and applications* (New York, NY, USA, 1999), SMA '99, ACM, pp. 270–277.

[10] Guibas, L., and Stolfi, J. Primitives for the manipulation of general subdivisions and the computation of voronoi. *ACM Trans. Graph. 4*, 2 (Apr. 1985), 74–123.

[11] Hatcher, A. *Algebraic topology*. Cambridge University Press, 2002.

[12] Hoffmann, C. M., and Kim, K.-J. Towards valid parametric cad models. *Computer-Aided Design 33*, 1 (2001), 81–90.

[13] Kalay, Y. E. The hybrid edge: a topological data structure for vertically integrated geometric modelling. *Comput. Aided Des. 21*, 3 (Apr. 1989), 130–140.

[14] Lee, S. H., and Lee, K. Partial entity structure: a compact non-manifold boundary representation based on partial topological entities. In *Proceedings of the sixth ACM symposium on Solid modeling and applications* (New York, NY, USA, 2001), SMA '01, ACM, pp. 159–170.

[15] Lienhardt, P. Topological models for boundary representation: a comparison with n-dimensional generalized maps. *Comput. Aided Des. 23*, 1 (Feb. 1991), 59–82.

[16] Mantyla, M. *Introduction to Solid Modeling*. W. H. Freeman & Co., New York, NY, USA, 1988.

[17] Paoluzzi, A., Bernardini, F., Cattani, C., and Ferrucci, V. Dimension-independent modeling with simplicial complexes. *ACM Trans. Graph. 12*, 1 (Jan. 1993), 56–102.

[18] Paoluzzi, A., Ramella, M., and Santarelli, A. Boolean algebra over linear polyhedra. *Comput. Aided Des. 21*, 10 (Oct. 1989), 474–484.

[19] Pascucci, V., Ferrucci, V., and Paoluzzi, A. Dimension-independent convex-cell based HPC: representation scheme and implementation issues. In *Proceedings of the third ACM Symposium on Solid Modeling and Applications* (New York, NY, USA, 1995), SMA '95, ACM, pp. 163–174.

[20] Pratt, M. J., and Anderson, B. D. A shape modelling api for the STEP standard. In *in Fourteenth International Conference on Atomic Physics* (1994), pp. 1–7.

[21] Raghothama, S., and Shapiro, V. Boundary representation deformation in parametric solid modeling. *ACM Trans. Graph. 17*, 4 (Oct. 1998), 259–286.

[22] Raghothama, S., and Shapiro, V. Consistent updates in dual representation systems. In *Proceedings of the fifth ACM symposium on Solid modeling and applications* (New York, NY, USA, 1999), SMA '99, ACM, pp. 65–75.

[23] Requicha, A. G. Representations for rigid solids: Theory, methods, and systems. *ACM Comput. Surv. 12*, 4 (Dec. 1980), 437–464.

[24] Rossignac, J. R., and O'Connor, M. A. SGC: a dimension-independent model for pointsets with internal structures and incomplete boundaries. In *Geometric modeling for product engineering: selected and expanded papers from the IFIP WG 5.2/NSF Working Conference on Geometric Modeling, Rensselaerville, U.S.A., 18-22 September, 1988*. North-Holland, 1990.

[25] Rossignac, J. R., and Requicha, A. A. G. Constructive non-regularized geometry. *Comput. Aided Des. 23*, 1 (Feb. 1991), 21–32.

[26] Shapiro, V. Solid modeling. In *Handbook of Computer Aided Geometric Design*, G. Farin, J. Hoschek, and S. Kim, Eds. Elsevier Science, 2002, ch. 20, pp. 473–518.

[27] Shapiro, V., and Vossler, D. L. What is a parametric family of solids? In *Proceedings of the third ACM symposium on Solid modeling and applications* (New York, NY, USA, 1995), SMA '95, Acm, pp. 43–54.

[28] Weiler, K. Edge-based data structures for solid modeling in curved-surface environments. *Computer Graphics and Applications, IEEE 5*, 1 (jan. 1985), 21 –40.

[29] Woo, T. A combinatorial analysis of boundary data structure schemata. *Computer Graphics & Applications, IEEE 5*, 3 (March 1985), 19–27.

[30] Yamaguchi, Y., and Kimura, F. Nonmanifold topology based on coupling entities. *Computer Graphics and Applications, IEEE 15*, 1 (1995), 42–50.