# Topology Computing w GraphBLAS Sparse Matrix Algebra

Franco Milicchio[a], Alberto Paoluzzi[b], Gianmaria DelMonte[a], and Omar Elsayed[a]
Department of {Engineering[a], Mathematics and Physics[b]}, Roma Tre University, Rome, Italy

February 21, 2020

## Contents

**1 Introduction** **1**

**2 Background: Chain Complexes** **3**
2.1 Cell complexes vs Chain complexes . . . . . 3

**3 Chain Adjacency and Incidence** **4**
3.1 Boundary representation of solid models . . . . . 4
3.2 Space Decompositions . . . . . 4
3.3 Minimal representations . . . . . 5

**4 Sparse Matrices and Solid Boolean Algebras** **5**

**5 Cellular Complexes with GraphBLAS** **6**
5.1 Geometric / topological sparse matrices . . . . . 7
5.2 Matrix computation of boundary chain . . . . . 7
5.3 GraphBLAS computation of boundary chain . . . . . 7

**6 Simple examples** **8**

**7 Conclusion** **9**

## 1 Introduction

The aim of this paper is to introduce an approach to 2D and 3D computational topology and geometry using a *GraphBLAS* sparse matrix representation of *chain complexes* with linear operators $\partial_p$ and $\delta_{p-1} = \partial_p^\top$ between linear *chain* spaces $C_p$:

$$C_\bullet = (C_p, \partial_p) := C_3 \underset{\partial_3}{\overset{\delta_2}{\leftrightarrows}} C_2 \underset{\partial_2}{\overset{\delta_1}{\leftrightarrows}} C_1 \underset{\partial_1}{\overset{\delta_0}{\leftrightarrows}} C_0.$$

Let us to remark preliminary that when $p \in \{0, 1\}$, a chain complex is a representation of a *graph*, with $[\partial_1]$ the *incidence matrix* between 1-cells (edges) in $C_1$ and 0-cells (vertices) in $C_0$. Also, the multiplication $[\partial_1][\delta_0] = [\partial_1][\partial_1]^t$ gives the *adjacency* matrix of the graph, while the diagonal entries provide the degrees of vertices, defined by the number of incident edges on each vertex. The *GraphBLAS* standard [?] provides a small set of matrix primitives to compute graph properties that, combined together, allow for easy implementation of fast algorithms on large graphs.

In several areas of geometric and topological computing—including geo-mapping, building information modeling, medical imaging, CAD and solid modeling, virtual and augmented reality, finite element modeling and simulation, etc.—the amount and detail of 2D and/or 3D data continue to grow. Analogously, the need for an unified approach to graph algorithms and for unified and simplified interfaces, has been well intercepted by the GraphBLAS initiative and the related GraphBLAS standardization effort `graphblas.org`. We would like to show here that the domain covering of this standard library on graphs can be greatly extended, to cover the representation of more general *cellular d-complexes* $(1 \leq d \leq 3)$ and their $p$-skeletons $(0 \leq p \leq 3)$.

(Co)chain complexes, as well (co)boundary operators, are well-known basic tools of algebraic topology and homological algebra. In particular, a *chain complex* is a graded sequence of such linear operators between graded linear spaces of "chains". A $p$-chain can be seen as a subset of a finite set `V` of Euclidean points whose affine hull has dimension $p$. Chain spaces and sparse matrices are the components of the Linear Algebraic Representation (`LAR`) [?], that is being used for boundary, decompositive, and enumerative representations [?] of models of rigid solid objects.

Some numerical methods aiming to integrate domain modeling, differential topology and mathematical modeling with physical simulations were based on chains and cochains, starting with [?, ?]. In particular, Discrete Exterior Calculus (DEC) with simplicial complexes was introduced by [?] and made popular by [?, ?]. Finite Element Exterior Calculus (FEEC) is an advance in the mathematics of finite element methods [?, ?, ?] that employs differential complexes to construct stable numerical schemes. The Cell Method (CM) is a purely algebraic computational method for modeling and simulation [?, ?, ?] based on boundary/coboundary maps and a direct discrete formulation of field laws. Our own research in geometrical and physical modeling with chain and cochain complexes was introduced in [?, ?, ?].

More recently, we provided—using sparse matrices—an algorithmic pipeline [?] to compute the *arrangement* of the Euclidean $d$-space $(d = 2, 3)$, i.e. the partition of it into a cellular $d$-complex, starting from a collection of (possibly intersecting) cellular ($d$-1)-complexes embedded in $\mathbb{E}^d$. In [?] we have shown that the *atoms* of the Boolean algebra generated by such space partition correspond one-to-one to the columns of the $[\partial_d]$ matrix of the boundary operator $\partial_d : C_d \to C_{d-1}$. This allows for fast native reconstruction of every solid expression in the *solid algebra* of $d$-space generated by the input terms, usually called Constructive Solid Geometry (CSG) in solid modeling [?].

The topological background of linear chain spaces and chain complexes is summarized in Section 2, together with the concepts of boundary and coboundary linear maps between chain spaces. In Section 3 an operational definition of topological queries through composition of operators corresponding to products of their sparse matrices is discussed. In Section 4 an interpretation as a Boolean algebra of the partition of the Euclidean $d$-space produced by a collection of geometric objects is provided. The implementation of such concepts using GraphBLAS is given in Section 5. Some simple examples of topology computation using linear algebra are shown in Section 6. In the Conclusion Section we briefly summarize our main points, state our findings, and propose some possible extension of this approach.

## 2 Background: Chain Complexes

Cellular complexes are largely used in Computer Graphics and in Geometric and Solid Modeling [**?**]. In particular, they provide the geometric-topological discretization for computer modeling and simulation of physical properties of both manmade and natural objects [**?**, **?**].

### 2.1 Cell complexes vs Chain complexes

A $p$-chain can be seen as a subset of $p$-cells from a cellular complex. The space of $p$-chains is closed w.r.t. addition and product times a scalar from a field. In particular, it is a linear (vector) space.

**Cells and Chains** A *$p$-manifold* is a topological space where each point has a neighborhood that is homeomorphic to $\mathbb{E}^p$. A *$p$-cell* $\sigma$ $(0 \leq p \leq d)$ of cellular complexes is piecewise-linear, connected, possibly non convex, $p$-manifold, and not necessarily contractible[1]. An $r$-face $\tau$ of a $p$-cell $\sigma$ $(0 \leq r \leq p)$ is an $r$-cell contained in the frontier of $\sigma$.

A *$p$-chain* can be seen, with some abuse of language, as a collection of $p$-cells. The set $C = \oplus\, C_p$ of chains can be given the structure of a graded vector space by defining sums of chains with the same dimension, and products times scalars in a field, with the usual properties.

A *basis* $U_p$ is the set of *independent* (or *elementary*) chains $u_p \in C_p$, given by singletons of $\Lambda_p$ elements. Every chain $c \in C_p$ is uniquely generated by a linear combination of the basis with field coefficients. Once the basis is fixed, the coordinate representation of each $\{\sigma_k\} = u_k \in C_p$ is unique. This is an ordered sequence of coefficients, either from $\{0, 1\}$ (unsigned representation) or from $\{-1, 0, +1\}$ (signed representation). With abuse of language, we often call $p$-cells the independent generators of $C_p$, i.e. the elements of $U_p$.

**Chain and cochain complexes** A *graded vector space* is a vector space $V$ expressed as a direct sum of spaces $V_k$ indexed by integers in $[0, d]$:

$$V = \oplus_{k=0}^{d} V_k, \qquad [0, d] := \{k \in \mathbb{N} \mid 0 \leq k \leq d\}.$$

A linear map $f : V \to W$ between graded vector spaces is called a *graded map* of degree $p$ if $f(V_k) \subset W_{k+p}$.

A *chain complex* is a graded vector space $V$ furnished with a graded linear map $\partial : V \to V$ of degree $-1$ which satisfies $\partial^2 = 0$, called *boundary operator*. In other words, a chain complex is a sequence of vector spaces $C_k$ and linear maps $\partial_k : C_k \to C_{k-1}$, such that $\partial_{k-1} \circ\, \partial_k = 0$.

A *cochain complex* is a graded vector space $V$ furnished with a graded linear map $\delta : V \to V$ of degree $+1$ which satisfies $\delta^2 = 0$, called *coboundary operator*. That is to say, a cochain complex is a sequence of vector spaces $C^k$ and linear maps $\delta^k : C^k \to C^{k+1}$, such that $\delta^{k+1} \circ\, \delta^k = 0$.

Since any linear map $L : V \to W$ between linear spaces induces a dual map $L' : W' \to V'$ between their duals, any chain complex is associated with a dual cochain complex, and viceversa:

$$(\delta^k \omega)g = \omega(\partial_{k+1} g), \qquad \omega \in C^k, g \in C_{k+1}.$$

In a Euclidean space, chain and cochain spaces can be trivially identified [**?**], so that we use the $C_p$ notation for both spaces, with $\partial_p : C_p \to C_{p-1}$, and $\delta_p = \partial_{p-1}^\top : C_{p-1} \to C_p$.

[1]The cells of CW-complexes are contractible to a point. With our LAR representation they may contain internal holes.

## 3 Chain Adjacency and Incidence

Boundary decompositions (B-reps) are the typical representations used in solid modeling and computer graphcs. The model boundary is partitioned into vertices (`V`: 0-cells), edges (`E`: 1-cells), and faces (`F`: 2-cells), where faces are often triangles or more general convex cells.

### 3.1 Boundary representation of solid models

In particular, a typical solid modeling representation employs some specialized data structure to efficiently traverse the boundary, moving from some element to the adjacent ones (with same dimension) or to incident ones (with different dimension). Several data structures have been used for this purpose [**?**], taking into account both the efficiency of topological queries and the storage compactness. Three binary adjacency relations and six binary incidence relations can be needed by algorithms, as shown in Table 1a.

Using chain operators, or better their matrices, when an ordering of elements has been fixed inside the sets `V`, `E`, and `F` using one-dimensional arrays, such binary relations can be represented and/or computed as shown in Table 1b. It is easy to show (see Table 1b), that all linear operators between chain spaces corresponding to binary relations between boundary elements can be derived from $\partial_1$ and $\partial_2$ via matrix transposition or product over semirings [**?**, **?**].

Table 1: (a) The 9 binary relations between boundary elements, and the 16 binary relations between decompositive elements; (b) corresponding linear operators between chain spaces. Remember that $\partial_p : C_p \to C_{p-1}$ and that $\delta_p = \partial_{p+1}^\top$.

| | V | E | F | C |
|---|---|---|---|---|
| V | VV | VE | VF | VC |
| E | EV | EE | EF | EC |
| F | FV | FE | FF | FC |
| C | CV | CE | CF | CC |

| | $C_0$ | $C_1$ | $C_2$ | $C_3$ |
|---|---|---|---|---|
| $C_0$ | $[\partial_1][\delta_0]$ | $[\partial_1]$ | $[\partial_1][\partial_2]$ | $[\partial_1][\partial_2][\partial_3]$ |
| $C_1$ | $[\delta_0]$ | $[\delta_0][\partial_1]$ | $[\partial_2]$ | $[\partial_2][\partial_3]$ |
| $C_2$ | $[\delta_1][\delta_0]$ | $[\delta_1]$ | $[\delta_1][\partial_2]$ | $[\partial_3]$ |
| $C_3$ | $[\delta_2][\delta_1][\delta_0]$ | $[\delta_2][\delta_1]$ | $[\delta_2]$ | $[\delta_2][\partial_3]$ |

Note that we have assumed vectors as column matrices, so that an operator maps the column space to the row space of its matrix. Hence, in order to maintain the consistency between the two representations we should read the relation $\texttt{AB} \subset \texttt{A} \times \texttt{B}$ as a map $\texttt{AB} : \texttt{B} \to \texttt{A}$.

### 3.2 Space Decompositions

A similar representation scheme—called *decompositive* by [**?**]—is also used, both in solid modeling and in order to represent the domain decomposition in FEM and other discretizations of physical models. In this case a partition of the whole domain is provided, using cellular 3-complexes, given by `V`, `E`, `F`, and `C` sets of 0-, 1-, 2- and 3-cells, where often the `C` elements are either tetrahedra or hexahedra.

For the topology of a graph (cellular 1-complex) just $C_0$ (vertices) and $C_1$ (edges) are needed, so that a complete representation is given by the signed or unsigned matrix $[\partial_1]$, corresponding to the relation $\texttt{EV} \equiv C_1 \to C_0$.

It is worth noting that operators $\partial_1$, $\partial_2$, and $\partial_3$ are sufficient to represent the complete collection of $4\times4$ operators (see Table 1b), via matrix transposition or product over semirings. In other words, the *chain complex* $C_\bullet = (C_p, \partial_p)$ of boundary matrices, $1 \leq p \leq 1, 2, 3$ is a *complete representation* of the topology of (a) graphs, (b) B-reps, and (c) decompositive representations, respectively.

Cellular complexes may be either *oriented* (signed) or *non-oriented* (unsigned). In the first case the elements of operator matrices are taken over the domain $D = \{0, 1\}$; in the second case they belong to the domain $D = \{-1, 0, 1\}$, so that passing from a coefficient $+1$ to a coefficient $-1$ (or viceversa) the element orientation is reversed.

## 3.3 Minimal representations

Most of earlier algorithms and procedures [**?**, **?**, **?**, **?**, **?**, **?**, **?**, **?**, **?**, **?**, **?**, **?**, **?**, **?**, **?**, **?**, **?**, **?**, **?**, **?**, **?**, **?**, **?**, **?**, **?**, **?**, **?**, **?**, **?**, **?**, **?**, **?**, **?**, **?**, **?**, **?**, **?**, **?**, **?**, **?**, **?**, **?**, **?**, **?**, **?**] work with data structures optimized for selected classes of geometric objects. By contrast, our formulation, representation, and algorithms, cast in terms of (co)chain complexes of (co)boundary maps, may be applied to very different geometric objects, ranging from solid models to engineering meshes, geographical systems, biomedical images.

Given a set $S = \{s_j\}$, the *characteristic function* $\chi_A : S \to \{0, 1\}$ takes value 1 for all elements of $A \subseteq S$ and 0 at all elements of $S$ not in $A$. We call *characteristic matrix* $M$ of a collection of subsets $A_i \subseteq S$ $(i = 1, \ldots, n)$ the binary matrix $M = (m_{ij})$, with $m_{ij} = \chi_{A_i}(s_j)$. A matrix $M_p$, whose rows are indexed by unit $p$-chains and columns are indexed by unit 0-chains, provides a useful representation of a basis for the linear space $C_p$. Permuting (reindexing) either rows or columns provides a different basis. While chains are mostly presented as formal sums of cells, in the actual implementation their signed coordinate vectors are used as *sparse* arrays, and in particular as CSC (Compressed Sparse Column) maps : $\mathbb{N} \to \{-1, 0, 1\}$.

It is possible to show [] that, when the $d$-cells are convex, the topology of a cellular $d$-complex is fully described by $M_{d-1}$ and by an embedding function $\mu : \mathtt{V} \to \mathbb{E}^d$. When $d$-cells are more complex—say, non-convex or with holes—the triple $(M_{d-1}, M_{d-2}, \mu)$ is needed to get a full knowledge of the topology of the complex.

We call `LAR` (Linear Algebraic Representation) of subsets of a set `V` (vertices), the array, indexed by ordinals (one-to-one with subsets), of arrays of indices to `V` elements. This one is a compact representation of the *characteristic matrix* of the collection of subsets.

# 4 Sparse Matrices and Solid Boolean Algebras

Manuscript [**?**] shows that the set of join-irreducible atoms of the Boolean Algebra generated by a partition of $\mathbb{E}^3$ are one-to-one with the basis of 3-chain space also generated by the partition, and hence with the columns of the $[\partial_3]$ matrix, which provides a boundary representation (as 2-cycles) of the independent elements of 3-space. An example of such space decomposition is shown in Figures 1 and 2. The 3-cells are not in scale, and are suitably rotated to better exhibit their complex structure, even containing internal holes. Their assembly gives the union of the five cubes. Each *3-cell is generated* as *a column* of the *sparse matrix* of *chain map* $\partial_3 : C_3 \to C_2$, with values in $\{-1, 0, 1\}$. They are the join-irreducible *atoms* of the CSG algebra with closed regular cells.

To generate the $d$-space arrangement induced by a collection of cellular ($d$–1)-complexes, some numerical algebra and basic tools of linear algebra and algebraic topology are used. In particular, (sparse) matrices of operators and matrix multiplication and transposition. Interval-trees and *kd*-trees are also introduced for acceleration of clustering of face cells into subsets of congruent shape.

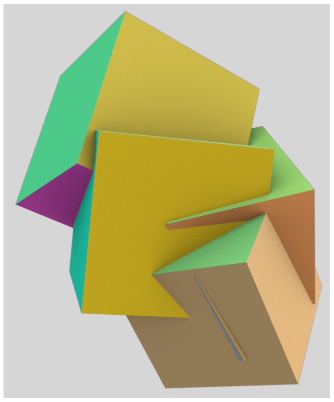
Figure 1: A collection $\mathcal{S}$ of five random cubes in 3D Euclidean space.

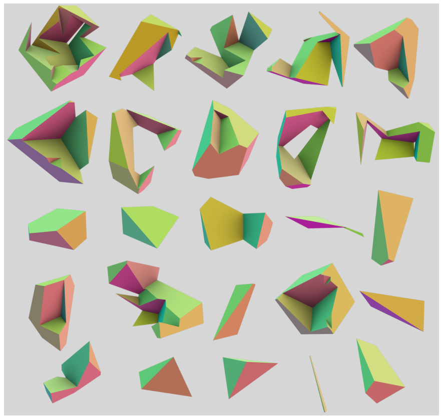
Figure 2: 3-cells of the generated arrangement $\mathcal{A}(\mathcal{S})$ of $\mathbb{E}^3$. They are given by columns of $[\partial_3]$ as *atoms* of a solid Boolean algebra.

Using our approach with sparse matrices, the validity of topological computations is guaranteed, since these operator matrices must satisfy by construction the (graded) constraints: $[\partial_2][\partial_3] = [0]$ and $[\partial_1][\partial_2] = [0]$. Similarly, we have $[\delta_1][\delta_0] = [0]$ and $[\delta_2][\delta_1] = [0]$.

With our approach based on boundary operators, the distinctions are removed between manifold and non-manifold representations (fairly standard in Solid Modeling), so allowing for mixing B-reps, cellular decompositions of elementary solids, and/or regular grids. This allows, e.g., for the generation of internal structures and mixed-dimensional objects needed in many applications, for example to make stronger and more resilient the ones to be produced via 3D printing.

Even more, the evaluation of CSG expressions of arbitrary complexity is done in a novel way, by combining the binary columns of 3D basis elements (i.e., the elements of the $U_3$ basis) with native Julia's operators for bitwise operations. In other words, once the 3-space partition is generated, and 3-cells are classified w.r.t. all solids terms, via a single point-set containment test, *all CSG algebraic expressions*—of any complexity—can be evaluated simply by bitwise vectorized logical operations. Finally, the sparse matrix approach can be extended to general dimensions and/or implemented on highly parallel computational engines, even using standard GPU computing kernels.

## 5 Cellular Complexes with GraphBLAS

We have implemented in Julia [?]—the novel language for scientific computing—our topological operations over cell complexes, using the package `SuiteSparseGraphBLAS.jl`, which is a Julia wrapper [?] for `SuiteSparse:GraphBLAS`, i.e., the GraphBLAS standard [?, ?, ?] provided within the *SuiteSparse* library [?] of sparse matrix software.

### 5.1 Geometric / topological sparse matrices

Within a typical computational pipeline in geometric applications, we may distinguish at least three types of sparse matrices: (a) characteristic matrices of cells as subsets of vertices; (b) boundary representations of edges, faces and solid cells; (c) matrix representation of binary incidence/adjacency relations between cells of different dimension.

**Characteristic matrices** provide the simplest representation of the independent elements (i.e., those that cannot be generated by linear combination of other elements) of a $p$-chain space $C_p$ $(0 \leq p \leq d)$ from a cellular $d$-complex $(2 \leq d \leq 3)$. They are built from arrays of arrays of vertex indices using the sol-called “coords” method, i.e., starting from $(i, j, x)$ triples.

**Boundary operators** give a simple mathematical representation of the so-called “B-reps”, normally used for solid models, in our case extended to cells of every dimension. In particular, every column of the $[\partial_p]$ matrix gives the signed representation of a basis $p$-cell (i.e., an independent $p$-chain) as a ($p$-1)-cycle (i.e., a ($p$-1)-chain without-boundary). They are built by multiplication of two characteristic matrices, followed by suitable “filtering” of values.

**Incidence relations** are generated by multiplication of the characteristic matrices of the two types of cells (of dimension $p$ and $q$, say) under consideration. The non-zero $(i, j)$ element of the product matrix provides the “strength” of the elementary incidence, i.e., the number of vertices shared between the $i$-th $p$-cell and the $j$-th $q$-cell. Their building is done by multiplication of two appropriate instances of (co)boundary matrices, as shown in Table 1b.

## 5.2 Matrix computation of boundary chain

The mathematics used to compute the graded chain complex $C_\bullet = (C_p, \partial_p)$ starting from sparse binary characteristic matrices $M_p$, with $p$-cells indexing the rows and 0-cells indexing the columns is given below. The boundary matrices $\partial_p$ $(1 \leq p \leq 3)$ between non-oriented chain spaces are computed by *sparse matrix multiplication* of characteristic matrices, followed by *matrix filtering*, produced in Julia by broadcasting vectorized integer division, i.e., “ .÷”, as follows:

$\partial_1$ = $\mathtt{M}_0 * \mathtt{M}_1'$ = $\mathtt{M}_1'$
$\partial_2$ = $(\mathtt{M}_1 * \mathtt{M}_2')$ .÷ sum($\mathtt{M}_1$,dims=2)
$\partial_3$ = $(\mathtt{M}_2 * \mathtt{M}_3')$ .÷ sum($\mathtt{M}_2$,dims=2)

## 5.3 GraphBLAS computation of boundary chain

Our current open-source implementation of cellular and chain complexes using sparse matrices and `SuiteSparseGraphBLAS.jl` is mantained in `https://github.com/gmgigi96/SparseMM`, where it provides fast and easy-to-use matrix tools for geometric and topological computing, including the input of cellular complexes, the computation of (unsigned) boundary operators, the answer to single and multiple queries about incidence or adjacency of cells.

# 6 Simple examples

Without loss of generality, let us start with the B-rep of a unit cube (see Figure **??**a) whith topology given by two arrays of arrays `EV` and `FV` providing indices of vertices `V` on boundary of edges `E` or faces `F`.

```
EV = [[1,2],[3,4],[5,6],[7,8],[1,3],[2,4],[5,7],[6,8],[1,5],[2,6],[3,7],[4,8]]
FV = [[1,2,3,4],[5,6,7,8],[1,2,5,6],[3,4,7,8],[1,3,5,7],[2,4,6,8]]
```

The $[\partial_1]$ matrix is ready to compute from `EV`:

```
n = length(EV);
Is,Js,Vs = map(cat,[EV, [[i,i] for i=1:n], [[1,1] for i=1:n]]);
∂₁ = sparse(Is,Js,Vs);

Matrix(convert(SparseMatrixCSC{Int8,Int64}, ∂₁))
8x12 Array{Int8,2}:
 1  0  0  0  1  0  0  0  1  0  0  0
 1  0  0  0  0  1  0  0  0  1  0  0
 0  1  0  0  1  0  0  0  0  0  1  0
 0  1  0  0  0  1  0  0  0  0  0  1
 0  0  1  0  0  0  1  0  1  0  0  0
 0  0  1  0  0  0  0  1  0  1  0  0
 0  0  0  1  0  0  1  0  0  0  1  0
 0  0  0  1  0  0  0  1  0  0  0  1
```

The $\partial_2$ matrix is computed by filtering the elements from the product $[\partial_1^\top] *$ `fv'`, where the sparse matrix `fv` is generated from `FV` array.

```
m = length(FV);
Is,Js,Vs = map(cat,[[[i for k=1:length(f)] for (i,f) in enumerate(FV)],
    [FV[i] for i=1:m], [ones(Int8, length(FV[i])) for i=1:m]]);
fv = sparse(Is,Js,Vs);
Matrix(convert(SparseMatrixCSC{Int8,Int64}, fv))
6x8 Array{Int8,2}:
 1  1  1  1  0  0  0  0
 0  0  0  0  1  1  1  1
 1  1  0  0  1  1  0  0
 0  0  1  1  0  0  1  1
 1  0  1  0  1  0  1  0
 0  1  0  1  0  1  0  1

triples = map(tuple,SparseArrays.findnz(∂₁' * fv')...);
mat3xm = hcat([ [i,j,1] for (i,j,v) in triples if v==2]...);
Is,Js,Vs = [mat3xm[1,:], mat3xm[2,:], convert(Array{Int8,1},mat3xm[3,:])];
∂₂ = sparse(Is,Js,Vs)
Matrix(∂₂)
12x6 Array{Int64,2}:
 1  0  1  0  0  0
 1  0  0  1  0  0
 0  1  1  0  0  0
 0  1  0  1  0  0
 1  0  0  0  1  0
 1  0  0  0  0  1
 0  1  0  0  1  0
 0  1  0  0  0  1
 0  0  1  0  1  0
 0  0  1  0  0  1
```

```
0  0  0  1  1  0
0  0  0  1  0  1
```

VV = [[k] for k=1:size(V,2)] model = (V, [VV,EV,FV])::Lar.LARmodel
meshes = GL.numbering(1.5)(model, GL.COLORS[1], 0.1) GL.VIEW(meshes);

## 7 Conclusion

In this paper we have summarized the main points of our approach to geometric and solid computing with basic computational topology using sparse matrices, and have discussed the current implementatation of some related matrix operations with SuiteSparse:GraphBLAS in Julia.

In particular, we have shown (a) the construction of matrix representation of cellular complexes, (b) the building of graded boundary and coboundary matrices, to efficiently navigate within a cellular complex (c) the setup of maps between chain spaces, that are equivalent to database queries on boundary elements of a solid representation.

Currently we are implementing with GraphBLAS our algorithmic pipeline to both generate a space arrangement and/or evaluate any expressions of solid algebras. We are currently starting to evaluate the efficiency of this approach with large-scale cellular and simplicial complexes, like the ones used for CAD of very complex engineering objects and assemblies. Experiments with complex biological structures are also being developed.

We strongly believe that our use of sparse-matrix-based topology might probably be combined with tensor-based neural networks, in order to move beyond image understanding and towards full reconstruction of built environments and complex scenes, starting from multiple images.

## References