Arrangements of cellular complexes[☆]

Alberto Paoluzzi[a], Vadim Shapiro[b], Antonio DiCarlo[c]

[a] *Roma Tre University, Italy*
[b] *University of Wisconsin-Madison & ICSI, United States*
[c] *CECAM-IT-SIMUL Node, Italy*

Abstract

In this paper we introduce a novel algorithm to combine two or more cellular complexes, providing a minimal fragmentation of the cells of the resulting complex. This algorithm has several important applications, including Boolean operations over big geometric data, like the detailed geometric modeling of buildings, and the smooth combination of 3D meshes. The algorithm operates over the LAR representation of argument complexes, based on sparse matrices, so being well-suited for implementation on last generation accelerators and GPGPU applications.

Keywords: Computational topology, Solid modeling, Linear Algebraic Representation, Arrangements, Cellular Complexes.

1. Introduction

Given a finite collection $\mathcal{S}$ of cellular complexes in $\mathbb{E}^d$, $d \in \{2, 3\}$, the *arrangement* $\mathcal{A}(\mathcal{S})$ is the decomposition of $\mathbb{E}^d$ into connected open cells of dimensions $0, 1, \ldots, d$ induced by $\mathcal{S}$.

In this paper, we discuss the computation of the arrangement produced by a given set of cellular complexes in either 2D or 3D. Our goal is to provide a complete description of the plane or space decomposition induced by the input, into cells of dimensions 0, 1, 2 or 3.

A planar collection $\mathcal{S}$ may include line segments, open or closed polygonal lines, polygons, two-dimensional meshes, and discrete images in 2D. A space collection may include 3D polygons, polygonal meshes, B-reps of solid models—either manifold or non-manifold, three-dimensional CAE meshes, and volumetric images in 3D.

The result of the computation discussed in this paper is the arrangement $\mathcal{A}(\mathcal{S}) = X$, with $X := \bigcup_{k=0}^{d} X_k$, and usually with $d \in \{2, 3\}$, providing a 2D or 3D cellular decomposition of the space $\mathbb{E}^d$ embedding S.

For example, you may consider a set $\mathcal{L} = \{l_1, l_2, \ldots\}$ of line segments l_h in $\mathbb{E}^2$, with $1 \leq h \leq m$, and the plane arrangement generated by it (see [1]), i.e. the 2-dimensional complex made by open cells of dimension 0, 1 and 2. Analogously, you may consider a set $\mathcal{P} = \{p_1, p_2, \ldots\}$ of planar polygons p_h in $\mathbb{E}^3$, with $1 \leq h \leq n$. In this case the space arrangement $\mathcal{A}(\mathcal{P})$ is a 3-dimensional complex X made by open 0-, 1-, 2- and 3-cells, which is locally homogeneous (or regularized) by construction.

1.1. Problem statement

In general, we want to compute the (regularized) cellular d-complex X generated by a set of $(d-1)$-complexes of linear, bounded and connected cells. This problem can be identified with the construction, by induction, of a d-dimensional cellular complex, through the filtration (or stratification) of its *skeletons* $X_0, X_1, \ldots, X_{d-1}, X_d$. In fact, the unknown d-cells of the output arrangement are generated starting from 1-cells, which are used to compute the coboundary operator δ_1, used in turn to compute the 2-cells and the δ_2 operator, and so on, by iteratively growing in dimension.

The robustness of geometrical and topological computations is approached here by lowering the dimension of numerical computations every time it is possible, and by solving independently the resulting set of subproblems. For example, the intersection of 3-cells is reduced to a set of intersections of 2-cells, and each of those to the resolution of a set of segment intersections. The topology is finally reconstructed bottom-up, by successive identification of geometrical elements (reduction to quotient spaces) by nearest neighbourhood queries.

The problem studied in this paper has a number of useful geometric applications, including in particular the motion planning of robots and the variadic[1] computation of Boolean operations (union, intersection, difference, symmetric difference), provided a list of cellular complexes embedded in the same Euclidean space.

1.2. Previous work

The construction of arrangements of lines, segments, planes and other geometrical objects can be found in [2] together with a description of the CGAL software. A wide analysis of papers and algorithms concerning building and counting of cells may be found in the dedicated survey chapter on Arrangements in the Handbook of Discrete and Computational Geometry [3]. The arrangements of polytopes, hyperplanes and d-circles are discussed in [4]. At the best of our knowledge, no literature

[☆] Partially supported by a grant from SOGEI S.p.A. — the ICT company of the Italian Ministry of Economy and Finance.

[1] Which accepts a variable number of arguments.

exists for the computational problem discussed in this paper, written as a guide for a dedicated software library.

1.3. Paper preview

In the introduction section we have discussed the generalities of the construction of the space arrangement generated by a set of cellular complexes. In Section 2 the main definitions concerning CW-complexes and chain complexes are recalled, together with the characteristics of the Linar Algebraic Representation scheme. The main topic of the paper, i.e. the novel algebraic algorithm for computation of the merge of complexes, is discussed in Section 3. In Section 4 a couple of simple examples are presented. The conclusion section 5 makes a summary of the work and highlights its salient characteristics.

2. Background

2.1. Cellular complex and Chain spaces

Let X be a topological space, and $\Lambda(X) = \bigcup_k \Lambda_k$ ($k \in 0, 1, \ldots, d$) be a partition of X, with Λ_k a set of open k-cells. *CW-structure* on the space X is a filtration $\emptyset = X_{-1} \subset X_0 \subset X_1 \subset \ldots \subset X = \bigcup_d X_d$, such that, for each k, the *skeleton* X_k is homeomorphic to a space obtained from X_{k-1} by attachment of k-cells in $\Lambda_k = \Lambda_k(X)$.

CW-complex is a space X endowed with a CW-structure, and is also called a *cellular complex*. A cellular complex is *finite* when it contains a finite number of cells. A *regularized* d-complex is a complex where every k-cell ($k < d$) is contained in the boundary of a d-cell.

Let $(G, +)$ be a nontrivial abelian (i.e., commutative) group. A *p-chain* of X with coefficients in G is a mapping $c_p : X \to G$ such that, for each $\sigma \in X_p$, reversing a cell orientation changes the sign of the chain value:

$$c_p(-\sigma) = -c_p(\sigma).$$

Chain addition is defined by addition of chain values: if c_{p_1}, c_{p_2} are p-chains, then $(c_{p_1} + c_{p_2})(\sigma) = c_{p_1}(\sigma) + c_{p_2}(\sigma)$, for each $\sigma \in X_p$. The resulting group is denoted $C_p(X; G)$. When clear from the context, the group G is often left implied, writing $C_p(X)$.

Let σ be an oriented cell in X and $g \in G$. The *elementary chain* whose value is g on σ, $-g$ on $-\sigma$ and 0 on any other cell in X is denoted $g\sigma$. Each chain can then be written in a unique way as a (finite) sum of elementary chains.

Chains are often thought of as attaching orientation and multiplicity to cells: if coefficients are taken from the group $G = \{-1, 0, 1\}$, then cells can only be discarded or selected, possibly inverting their orientation.

Actually, the ability to add (and subtract) chains is not enough: the very idea of *mesh refinement* (and coarsening) requires chains to be also scaled. Hence the full linear structure is brought to bear (see [5]).

2.2. Linear Algebraic Representation

LAR (Linear Algebraic Representation) [6] is a representation scheme [7] for geometric and solid modeling. The domain of the scheme is provided by *cellular complexes*, while its codomain is that of *sparse matrices*. The topology in LAR is given just the binary characteristic matrix M_d of d-cells for polytopal complexes, or the pair M_d, M_{d-1} for more general (non convex) types of cells.

Such very sparse matrices are stored in memory using either the CSR (Compressed Sparse Row) or the CSC (Compressed Sparse Column) memory format [8]. The LAR polyhedral domain coincides with complexes of connected d-cells, even non-convex and/or including any number of holes.

The very general shape allowed for cells makes the LAR scheme notably appropriate for solid modeling of buildings and their components. E.g., the whole facade of a building can be described as a single 3-cell of its solid model. Also, the algebraic foundation of LAR allows not only for fast queries about incidence and adjacency of cells, but also to resolve—via fast SpMV computational kernels—the boundary extraction of any 3D subset of the building model.

It is worth noting that LAR provides a direct management of all subsets of cells and their physical properties trough the linear spaces of *chains* induced by the model partitioning, and their dual spaces of *cochains*.

The linear operators of *boundary* and *coboundary* between such linear spaces, suitably implemented by sparse matrices, directly supply, depending on the dimension of the mapped spaces, the discrete differential operators of *gradient*, *curl* and *divergence*, while their product gives the *Laplacian* [5].

2.3. Representation of a complex

The common representation of a d-complex in both commercial and academic systems is some (often complicated) data structure to incarnate its $(d-1)$-boundary. In some cases, like for the meshes discretising the space of a simulation, or when the solid scheme is decompositive [7], the target of the storage structure is the set of d-cells or their $(d-1)$-skeleton, or both.

The representation of either the d or the $(d-1)$ cells is normally supplemented by the storage of some subsets of the incidence relations between topological elements, usually paired with some ordering though linked lists of pointers.

Conversely, with the LAR scheme, only the characteristic functions of d-cells as vertex subsets are strictly necessary for polytopal complexes (examples include simplicial, cubical, and Voronoi complexes). For more general cell complexes, including non-convex and non-contractible cells, useful in some applications like the modelling of buildings, both the d and $(d-1)$-skeletons are needed [6]. In all cases, (co)boundary operators and every topological query are computable by sparse matrix product computational kernels implemented on GPUs.

3. Merge algorithm

The computation of the arrangement of $\mathbb{E}^d$ generated by a collection of d-complexes, here also called *merge of complexes*,

is discussed in this section using the mathematical language of chain complexes, i.e. the linear spaces of chains (of cells) and the linear (co)boundary maps between such spaces.

It is worth noting that the support space (point-set union) of the $(d-1)$-skeleton of the output complex equates the union of the analogous input subspaces. In other words, the generated complex is minimally fragmented. The reader should consider how much this differs from other dimension-independent approaches based on spatial indices like 2^d-trees or BSP-trees.

3.1. General problem description

Let d be the dimension of the embedding space $\mathbb{E}^d$, and

$$\mathcal{S}_d = \{ X^h,\ h \in H\}$$

a collection of d-complexes, and let us denote with $X^h_{d-1} \subset X_d$ the $(d-1)$-th skeleton of the h-th d-complex. We intend to compute the arrangement of $\mathbb{E}^d$ generated by $\mathcal{S}_d$, or, more precisely, the regularized d-complex $X = \mathcal{A}(\mathcal{S}_{d-1})$. For the sake of concreteness, the reader may assume $d = 3$.

3.1.1. Place the $(d-1)$-skeletons together

We start by considering the set $B = \bigcup_{h\in H} X^h_{d-1}$, combinatorial union of $(d-1)$-cells in $\mathcal{S}_d$, where H is a set indexing the input complexes.

3.1.2. Compute the spatial index $\mathcal{I} : B \to 2^B$

An *interval tree* is a data structure to hold intervals [9], that allows to efficiently find all intervals that overlap with any given interval or point. It is mainly used for windowing queries. We compute a spatial index $\mathcal{I} : B \to 2^B$, mapping each $(d-1)$-cell σ to the subset of $(d-1)$-cells τ such that $\mathrm{box}(\sigma) \cap \mathrm{box}(\tau) \neq \emptyset$, where $\mathrm{box}(\gamma)$ is the *containment box* of $\gamma \in B$, i.e. the minimal d-parallelepiped parallel to the Cartesian frame, and such that $\gamma \subseteq \mathrm{box}(\gamma)$. The $\mathcal{I}$ map is computed by set intersection of the output of d elementary queries on one-dimensional interval trees.

3.1.3. Compute the facet arrangments in $\mathbb{E}^{d-1}$

We compute the set $\mathcal{X}_{d-1}$ of *facet arrangements* $\mathcal{A}(\mathcal{S}_{d-1})$ induced in $\mathbb{E}^{d-1}$ by σ elements of B, fragmented by all the incident cells $\mathcal{I}(\sigma)$. To compute the arrangements, for each $\sigma \in B$, submanifold maps $M : \mathbb{E}^d \to \mathbb{E}^d$ are easily determined, sending σ to the subspace with implicit equation $x_d = 0$:

$$\mathcal{X}_{d-1} = \{ \mathcal{A}(\sigma \cup \mathcal{I}(\sigma)),\ \sigma \in B \}.$$

Accordingly, for each $\sigma \in B$:

1. compute an affine transformation M that map σ to the subspace $x_d = 0$,
2. consider the set $\mathcal{S}_{d-1} = M(\Sigma)$, with $\Sigma = \{\sigma\} \cup \mathcal{I}(\sigma)$,
3. compute the hyperplane arrangement $\mathcal{A}(\mathcal{S}_{d-1}) = X^\sigma_{d-1}$,
4. map back the resulting $(d-1)$-complex to $\mathbb{E}^d$, i.e. compute $M^{-1}(X^\sigma_{d-1})$.

All such $(d-1)$-complexes embedded in $\mathbb{E}^d$ are therefore accumulated in $\mathcal{X}_{d-1} = \{M^{-1}(X^\sigma_{d-1}), \sigma \in B\}$, represented as the proper LAR structure.

3.1.4. Assembling the output $(d-1)$-skeleton

A *k-d* tree, or *k-dimensional tree*, is a binary search tree used to organize a discrete set of points with k dimensions. *K-d* trees are very useful for range and nearest neighbour searches [10].

Here, a *k-d* tree is computed over the vertices of the collection of complexes $\mathcal{X}_{d-1}$ in order to identify in a single point all the vertices contained in the same ϵ-neighborhood, with small $\epsilon \in \mathbb{R}$. As a consequence of this reduction of the vertex set, we

- rewrite the representation of each $(d-1)$-cell using the new vertex indices;
- sort the vertex indices of each $(d-1)$-cell to identify and remove the duplicate $(d-1)$-cells.

The resulting $(d-1)$-complex in $\mathbb{E}^d$ is the $(d-1)$-skeleton X_{d-1} of the output complex $X = \mathcal{A}(\mathcal{S}_d)$ generated by the input collection $\mathcal{S}_d$. The LAR representation is also reduced in *canonical form*, where the vertex indices are sorted in each cell.

3.1.5. Extraction of d-cells from $(d-1)$-skeleton

At the beginning of this stage of the algorithmic reconstruction of the space arrangement induced by a collection $\mathcal{S}$ of cellular complexes, we have a *complete* [7] representation in $\mathbb{E}^d$ of the skeleton X_{d-1} of $\mathcal{A}(\mathcal{S})$.

Therefore, a coordinate representation of the coboundary $\delta_{d-1} : C_{d-1}(X) \to C_d(X)$ is incrementally constructed here, by computing one-by-one the d-cells in X, i.e the rows in δ_{d-1}. This construction will complete our reconstruction of $\mathcal{A}(\mathcal{S})$. It is possible to show that this computation can be executed in parallel, with a number of independent tasks that equates the number of d-cells in X.

Let us consider the boundary map $\partial_d : C_d \to C_{d-1}$, with $\partial_d = \delta^\top_{d-1}$. Each column in ∂_d is the representation, in the basis of C_{d-1}, i.e., as a linear combination of the cells in X_{d-1}, of one element of the C_d basis (i.e., of a d-cell in X_d).

The construction of d-cells is carried out by using two tools:

- the property that every $(d-1)$-cell in a d-complex is shared *at most* by *two* d-cells [11]. Notice that if the complex includes also the *exterior cell*, then the incidence relation between cells in X_{d-1} and in X_d holds exactly;
- the computation of the cyclic subgroups of permutations of $(d-1)$-cells about their common $(d-2)$-cell, which imply the existence of a *next* and $prev = next^{-1}$ bijections on such subsets.

Notice that cyclic subgroups δc, with $c \in C_{d-2}$

1. are one-to-one with cells in X_{d-2},
2. we have $next(-c) = next^{-1}(c)$.

Since we deal with *oriented* chain complexes, we normally want to build a *coherently oriented* cellular complex X from the arrangement $\mathcal{A}(\mathcal{S}_d)$, by imposing that every facet shared by two d-cells in X_d must have opposite orientations on them. This re-orientation of d-cells can be executed—if possible, because not all d-complexes are orientable—in a successive final stage, in time linear with the number of cells and the size of X.

3.1.6. d-cells are minimal $(d-1)$-cycles

A *p-cycle* is a closed p-chain in $C(X)$. A chain $c \in C(X)$ is said *closed* if $\partial c = \emptyset$. A formal algorithm for extraction of d-cells of X starting from $(d-1)$-cells in X_{d-1} is given here. In particular, we generate the representation of a basis d-cell (more exactly: of a singleton d-chain, element of a C_d basis) by computing a *closed* and *minimal* $(d-1)$-chain, that later can be coherently oriented with the other d-cells in the C_d basis.

Each such $(d-1)$-chain is closed, i.e. without boundary—in other words, it is a $(d-1)$-cycle—in order to guarantee the satisfaction of the rule $\partial\partial c = 0$ for every generated singleton chain $c \in C_d(X)$.

We say that a basis of d-chains is *minimal* (or canonical) when the sum of lengths of its elements as (nonzero) combinations of facets equates the double of the number of $(d-1)$-cells. In other terms, if we denote the cardinality of a p-basis as $|X_p|$, we have, for a canonical d-basis and the matrix $[\partial_d] = (a_{i,j})$ with $a_{i,j} \in \{0, 1\}$

$$\sum_{i=1}^{|X_{d-1}|} \sum_{j=1}^{|X_d|} a_{i,j} = 2|X_{d-1}|.$$

This property, well known in solid modeling, is normally represented [12] as a set of identities between the cardinalities of incidence relations between vertices, edges and faces in 3D boundary representations and/or graphs drawn on a 2-manifold.

The minimality of each basis d-cell is guaranteed by the repeated application of the map

$$\text{stripe} : C_{d-1} \to C_{d-1},$$

that takes as input a $(d-1)$-chain c, starting from an initial "seed" $(d-1)$-cell, and returns a "stripe" (see Figures 1c, 1e, 1f) of boundary manifold at every repeated application, until $\partial c = \emptyset$:

$$c \mapsto c \cup \text{next}(\partial c) = c \cup (\bigcup_{\tau \in \partial c} \text{next}((\delta\tau) \cap c)).$$

In words, a basis element for C_d, made by a minimal $(d-1)$-chain, is generated from X_{d-1} by (i) choosing a cell $\sigma \in X_{d-1}$ and setting $c := \{\sigma\}$, then (ii) by repeating $c := c \cup \text{stripe}(c)$, until (iii) $\partial c = \emptyset$.

3.1.7. Cyclic subgroups of the $\delta\partial c$ chain

We remark that in computing the representation $c \in C_{d-1}$ of a C_d basis element, we iterate over "stripes" — $(d-1)$-chains — of coherently oriented X_{d-1} cells extracted from the coboundary $\delta\partial c \in C_{d-1}$ of a cycle $\partial c \in C_{d-2}$, with $n = |\delta\partial c|$ and $m = |\partial c|$.

Let us consider the symmetric group S_n of permutations of $\delta\partial c$ elements. By linearity of δ, the group elements can be partitioned into m cyclic subgroups $R \subset X_{d-1}$ associated one-to-one to each element $\tau \in \partial c$, and represented as curved arrows in Figure 2, with $\sum_{k=1}^{m} |R_k| = |\delta\partial c| = n$.

It is easy to prove that each R_k contains one and only one c component, computable as $\sigma = R_k \cap c$, so that the generic *stripe* application returns m cells, each one generated either by $next(\sigma)(R_k)$ or by $prev(\sigma)(R_k)$, depending from the orientation of each "hinge" cell $\tau \in C_{d-2}$ with respect to the orientation of ∂c, codified by either $+1$ or -1 coefficients in the algebraic expression of ∂c.

The computation of the minimal boundary of a d-cell is shown for $d = 2$ in Example 2 and Figures 2 and 3. Of course, what said above also holds for higher values of d, and in particular in 3-space, where the cyclic subgroups are around edges.

3.1.8. Linear independence of the extracted cells

The $(d-1)$-cells generated in Section 3.1.6 are not linearly independent, since one of them, and in particular the external cycle, is computable as sum of the others [13]. To remove the external cycle from the basis requires a dedicated test, that may be either very simple of more expensive, depending on the characteristics of the complex. The simplest test requires the computation of the number of cells in each basis cycle. With the greatest probability, the longest description identifies the external cell. If such assumption cannot be trusted, it is necessary to compute the absolute value of the signed volume of cells [14, 15]. The one greatest in volume will be removed from the basis.

3.1.9. Lattice of non-intersecting boundary components

When the result of applying the boundary operator ∂_d to the set of basis d-cells is an unconnected set of $(d-1)$-cycles, such boundary components must be compared with each other, to determine their actual orientation and the possible relative containment. For this purpose an efficient point classification algorithm must be available, in order to compute the component containment lattice.

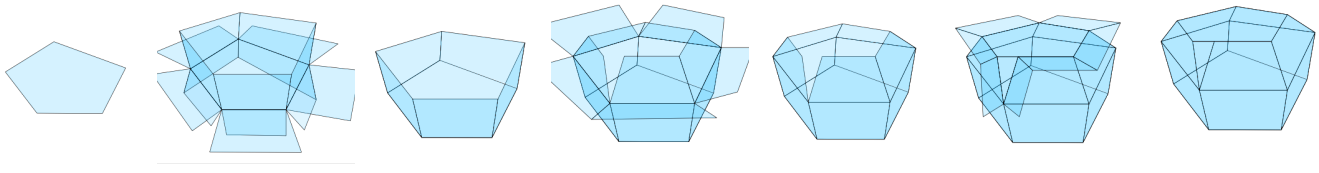
The $n \times n$ binary and antisymmetric matrix $M = (m_{i,j})$ of the containment relation between the external cycles of the n maximal 2-point-connected subgraphs (see Section 3.2.2) of each planar subproblem is constructed, computing each element of the matrix with a single point-cycle containment test, since the two corresponding cycles certainly do not intersect.

Then the global character of each cycle c_j as either external or internal (and hence its relative orientation) is given by the parity of $\sum_{p=1}^{n} m_{p,j}$. Further details on this point, whose discussion would require a long space, are left for the next paper.

3.2. Basic step: arrangement of lines in $\mathbb{E}^2$

As already said, the *Merge* algorithm *reduces the computation* of the arrangement of complexes $\mathcal{A}(\mathcal{S}_{d-1})$ in $\mathbb{E}^d$ to the computation of a set of independent problems in $\mathbb{E}^{d-1}$, and so on, *until to solve* independently *a set of problems* $\mathcal{A}(\mathcal{S}_1)$ in $\mathbb{E}^2$. The actual basic case in $\mathbb{E}^2$ is discussed in this section.

The reconstruction of the solution to the initial problem, i.e. the computation of $X = \mathcal{A}(\mathcal{S}_{d-1})$, is finally done by proper reduction to quotient sets. This reduction apply by (a) identification of coincident vertices through nearest neighbourhood queries on their k-d-tree, and (b) elimination of multiple *canonical* LAR instances of cells, which make use of sorted indices of vertices.



(a) (b) (c) (d) (e) (f) (g)

Figure 1: Extraction of a minimal 2-cycle from $\mathcal{A}(X_2)$: (a) the initial value for $c \in C_2$; (b) cyclic subgroups on $\delta\partial c$; (c) new value of c; (d) cyclic subgroups on $\delta\partial c$; (e) new value of c; (f) cyclic subgroups on $\delta\partial c$; (g) new value of c, such that $\partial c = 0$, hence stop.

3.2.1. Segment fragmentation: linear graph

Each input line segment is fragmented against all the intersecting segments, detected using a spatial index based on interval trees (Figure 4b). Each generated sub-fragment produces a pair of *unique* vertices. Quasi-coincident, i.e. numerically close, vertices are finally identified and a single 1-complex, stored as vertices and edges in a LAR object X_1, is created. The generation of subsegments is embarrassingly parallel.

3.2.2. Maximal 2-point-connected subgraphs

The X_1 complex is reduced to the union of its maximal 2-point-connected subgraphs, discovered by using the Hopcroft-Tarjan algorithm [16], in order to remove all the dangling edges and tree subgraphs. As a consequence, $\mathcal{A}(X_1)$ generates a partition of $\mathbb{E}^2$ with open, connected, and regularized, but yet unknown, 2-cells (Figure 4c) to be computed in the next step.

3.2.3. Topological extraction of X_2

The computation of the 1-chain representation of cells in X_2, providing also the signed ∂_2 operator, is discussed in Section 3.1.6, and shown in Figures 2 and 3. In Figures 4d and 4e is shown an extraction examples of 2-cell elements in X_2 from a random set of line segments.

3.3. Comments

The proposed approach is *embarrassingly parallel*, since no effort is needed to separate the problem into a number of independent parallel tasks. In particular, if B_2 is the collection of 2-cells in $\mathcal{S}$, and n_2 is their number, we have n_2 independent tasks for the computation of the planar arrangements X_2^σ, $\sigma \in B_2$.

Analogously, the extraction of 3-cells can be worked out in parallel, by independent tasks that compute a single closed 2-chain. The coherent orientation of the basis elements is very simple to perform sequentially a posteriori.

For example, given at least two complexes in $\mathbb{E}^3$, intersect all their polygonal facets with each possibly intersecting one, to produce the 0-, 1-, 2-, and 3-cells of their arrangement. Then iteratively order them to define 3-cells and (co)boundary operators. In this approach all geometric computations on each target facet are essentially two-dimensional, since the focus, after a proper affine map, is on partitioning the $z = 0$ plane against the affine hulls of incident polygons, i.e. against the line segments intersecting $z = 0$, so always reducing the problem to the arrangement of line segments (see Figure 4).

If computing entails only k-planes, e.g. lines in 2D and planes in 3D, then all cells in their arrangement are connected and convex [4], leading to straightforward geometric and topological computations. Conversely, if the arrangement is generated by intersection of line segments, or by possibly non convex plane or spatial cells, then the resulting cellular complex may lead to general (non convex) cells, possibly including holes, and to slightly more complicated algorithms for inclusion of hole vertices in their proper LAR cell, that we remember is just a sorted collection of vertex indices. For example, a CDT (Constrained Delaunay Triangulation) of every cell is needed in this case for correct computation of the permutation subgroups 3.1.7, and for visualization.

4. Applications and examples

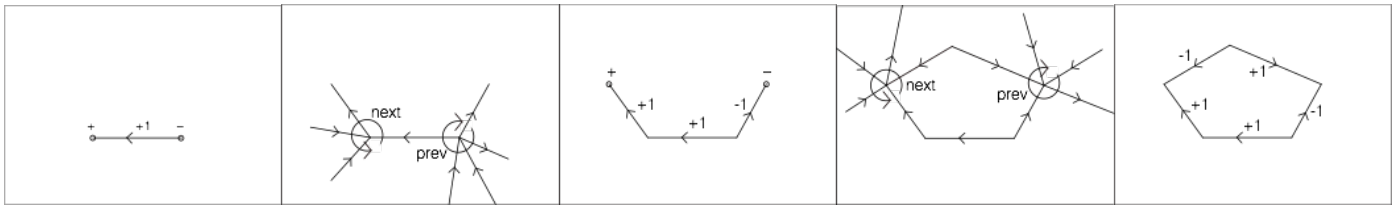
The first application of the *Merge* algorithm is for implementing Boolean operations between solids represented as cellular complexes, by using decompositions of either the boundary or the interior. For example, the free space for motion planning of robots in 2D or 3D can be obtained by merging the (grown) models of obstacles within a cubical mesh of the workspace, and by computing the generated arrangement.

As a further application, we are currently using chain complexes and boundary operators to extract the models of neurons and vessels from extreme-resolution 3D images of brain tissue, and to dramatically reduce the complexity of their representation, while preserving the homotopy type, in order to piecewise compute the connectome of brain structures.

Example 1. *An example of extraction of a minimal 2-cycle c, representation in C_2 of a basis element of C_3 is shown in Figure 1. The signs of the linear combination are usable as a matrix column of the operator $\partial_3 : C_3 \to C_2$.*

Let us preliminary recall some notions about p-chains as linear combinations of oriented $(p-1)$-chains, for $0 \leq p \leq d$. It is useful to select a conventional choice to orient the singleton chains (single cells) automatically. The 0-cells are considered all positive. The p-cells, for $1 \leq p \leq d-1$, can be given a coherent (internal) orientation *according to the orientation of the first $(p-1)$-cell in their* canonical *(sorted on facet indices) representation. Finally, a d-cell may be oriented as the sign of its oriented volume.*

It is possible to see that the ordering (numbering) of vertices, edges, and faces (i.e. of 0-, 1-, and 2-cells) completely determines the pattern of signs in the matrices of (c)boundary operators.



(a) (b) (c) (d) (e)

Figure 2: Extraction of a minimal 1-cycle from $\mathcal{A}(X_1)$: (a) the initial value for $c \in C_1$ and the signs of its boundary elements; (b) cyclic subgroups on $\delta\partial c$; (c) new (coherently oriented) value of c and signs of ∂c; (d) cyclic subgroups on $\delta\partial c$; (e) final value of c, with $\partial c = \emptyset$.

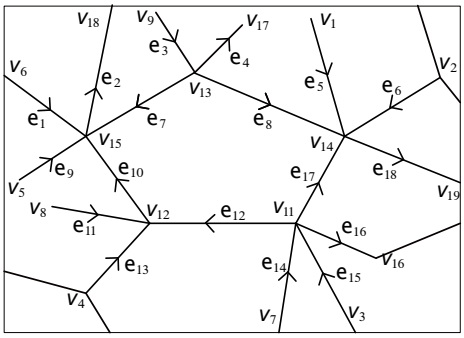


Figure 3: The portion of a 1-complex used by Example 2.

Example 2. *In Figure 3 we show a fragment of a 1-complex $X = X_1$ in $\mathbb{E}_2$, with cells $v_k \in X_0$ and $e_h \in X_1$. Here we compute stepwise the 1-chain representation $c \in C_1$ of a 2-cell of the complex $X_2 = \mathcal{A}(X_1)$. Look also at Figures 2a-e to follow stepwise the extraction of the 2-cell.*

(a) Set $c = e_{12}$. Then $\partial c = v_{12} - v_{11}$;

(b) then $\delta\partial c = \delta v_{12} - \delta v_{11}$ by linearity. Hence, $\delta\partial c = (e_{10} + e_{11} + e_{12} + e_{13}) - (+e_{12} + e_{14} + e_{15} + e_{16} + e_{17})$.

(c) Actually,by computing stripe(c) *we get*

$$\begin{aligned}\text{stripe}(c) &= c + \text{next}(c \cap \delta\partial c)\\ &= c + \text{next}(e_{12})(\delta v_{12}) - \text{next}(e_{12})(\delta v_{11})\\ &= e_{12} + \text{next}(e_{12})(\delta v_{12}) + \text{prev}(e_{12})(\delta v_{11})\\ &= e_{12} + e_{10} + e_{17}\end{aligned}$$

If we orient coherently c, we get $c = e_{10} + e_{12} - e_{17}$, and $\partial c = v_{15} - v_{12} + v_{12} - v_{11} + v_{11} - v_{14} = v_{15} - v_{14}$.

(d) As before, we repeat and reorient coherently the computed 1-chain:

$$\begin{aligned}\text{stripe}(c) &= c + \text{next}(c \cap \delta\partial c)\\ &= c + \text{next}(e_{10})(\delta v_{15}) - \text{next}(e_{17})(\delta v_{14})\\ &= e_{10} + e_{12} - e_{17} + \text{next}(e_{10})(\delta v_{15}) + \text{prev}(e_{17})(\delta v_{14})\\ &= e_{10} + e_{12} - e_{17} - e_7 + e_8\end{aligned}$$

(e) Finally, $\partial\,$stripe$(c) = \emptyset$ and the extraction algorithm terminates, giving $e_{10} + e_{12} - e_{17} - e_7 + e_8$ as a basis element for $C_2(X)$, with $X = \mathcal{A}(X_1)$, and hence as a column for the oriented matrix of the unknown $\partial_2 : C_2 \to C_1$.

5. Conclusion

In this paper we have introduced novel computational topology methods for discovering the space arrangement induced by a collection of cellular complexes. The best property of this approach consists in its minimal fragmentation of the output cells. There is no portion of the boundary of an output cell that was not present in the boundary cells of the input. The *Merge* algorithm also applies to any kind of complex of either the interior or the boundary of the merged spaces, including polytopal (e.g. Voronoi) complexes or complexes with non convex cells, even non contractible to a point.

The introduced methods differ significantly from other approaches not only because of the language, based on chain complexes and their operators, but also for the actual computer implementation, that makes only use of two classical data structures (k-d-tree of vertices and 1D interval trees) as computation accelerators, without inventing or using any fancy representation typical of non-manifold solid modeling, which might seem mandatory in this kind of geometric/topological algorithms and applications.

Conversely, the actual implementation, based on LAR [6] and implemented in Julia language [17] in a parallel computational environment using CUDA, efficiently describes cells and chains with either dense or sparse arrays of signed integers, and their linear operators with sparse matrices. It is being used both for deconstruction modeling of buildings [18] as well for the extraction of solid models of biomedical structures at the cellular scale [19, 20].

6. References

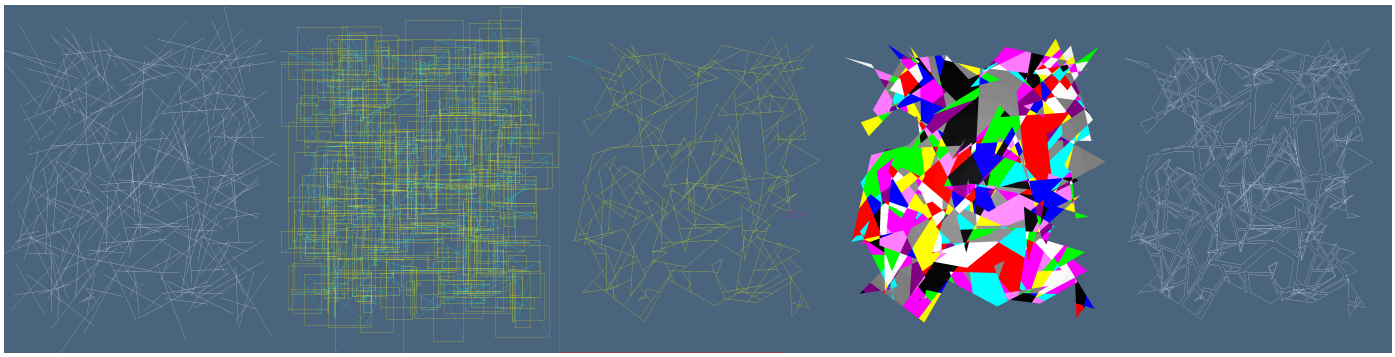
[1] Chazelle B, Edelsbrunner H, Guibas L, Sharir M, Snoeyink J. Computing a face in an arrangement of line segments and related problems. SIAM J Comput 1993;22(6):1286–302. URL: `http://dx.doi.org/10.1137/0222077`. doi:`10.1137/0222077`.

[2] Fogel E, Halperin D, Kettner L, Teillaud M, Wein R, Wolpert N. Arrangements. In: Boissonat JD, Teillaud M, editors. Effective Computational Geometry for Curves and Surfaces; chap. 1. Mathematics and Visualization; Springer; 2007, p. 1–66.

[3] Goodman JE, O'Rourke J, Tòth CD, editors. Handbook of Discrete and Computational Geometry – Third Edition. Boca Raton, FL, USA: CRC Press, Inc.; 2017. ISBN 0-8493-8524-5.

[4] Bjrner A, Ziegler GM. Combinatorial stratification of complex arrangements. J Amer Math Soc 1992;(5):105–49.

[5] DiCarlo A, Milicchio F, Paoluzzi A, Shapiro V. Chain-based representations for solid and physical modeling. Automation Science and Engineering, IEEE Transactions on 2009;6(3):454 –67. doi:`10.1109/TASE.2009.2021342`.



(a) (b) (c) (d) (e)

Figure 4: Computation of the plane arrangement $\mathcal{A}(\mathcal{L})$ generated by a set of line segments: (a) set of random lines in $\mathbb{E}^2$; (b) spatial index based on containment boxes; (c) plane partition $\mathcal{A}(X_1)$, where the 1-complex X_1 is the union of maximal 2-node-connected subgraphs in $\mathcal{A}(\mathcal{L})$, drawn here by yellow and cyan colour (up-left); (d) cells of X_2 in $X = \mathcal{A}(X_1)$, represented with random colours; (e) exploded boundaries of 2-cells of X_2.

[6] Dicarlo A, Paoluzzi A, Shapiro V. Linear algebraic representation for topological structures. Comput Aided Des 2014;46:269–74. URL: http://dx.doi.org/10.1016/j.cad.2013.08.044. doi:10.1016/j.cad.2013.08.044.

[7] Requicha AG. Representations for rigid solids: Theory, methods, and systems. ACM Comput Surv 1980;12(4):437–64. URL: http://doi.acm.org/10.1145/356827.356833. doi:10.1145/356827.356833.

[8] Buluç A, Gilbert JR. Parallel sparse matrix-matrix multiplication and indexing: Implementation and experiments. SIAM Journal of Scientific Computing (SISC) 2012;34(4):170 –91. URL: http://gauss.cs.ucsb.edu/~aydin/spgemm_sisc12.pdf. doi:10.1137/110848244.

[9] Preparata FP, Shamos MI. Computational Geometry: An Introduction. New York, NY, USA: Springer-Verlag New York, Inc.; 1985. ISBN 0-387-96131-3.

[10] Bentley JL. Multidimensional binary search trees used for associative searching. Commun ACM 1975;18(9):509–17. URL: http://doi.acm.org/10.1145/361002.361007. doi:10.1145/361002.361007.

[11] Hatcher A. Algebraic topology. Cambridge University Press; 2002. URL: http://www.math.cornell.edu/~hatcher/AT/ATpage.html.

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

[13] Mayeda W. Graph Theory. New York, NY: Wiley Interscience; 1972.

[14] Cattani C, Paoluzzi A. Boundary integration over linear polyhedra. Comput Aided Des 1990;22(2):130–5. URL: http://dx.doi.org/10.1016/0010-4485(90)90007-Y. doi:10.1016/0010-4485(90)90007-Y.

[15] Bernardini F. Integration of polynomials over n-dimensional polyhedra. Comput Aided Des 1991;23(1):51–8. URL: http://dx.doi.org/10.1016/0010-4485(91)90081-7. doi:10.1016/0010-4485(91)90081-7.

[16] Hopcroft J, Tarjan R. Algorithm 447: Efficient algorithms for graph manipulation. Commun ACM 1973;16(6):372–8. URL: http://doi.acm.org/10.1145/362248.362272. doi:10.1145/362248.362272.

[17] Bezanson J, Edelman A, Karpinski S, Shah VB. Julia: A fresh approach to numerical computing. SIAM Review 2017;59(1):65–98. URL: http://julialang.org/publications/julia-fresh-approach-BEKS.pdf. doi:10.1137/141000671. arXiv:http://dx.doi.org/10.1137/141000671.

[18] Marino E, Spini F, Salvati D, Vadalà C, Vicentino M, Paoluzzi A, et al. Modeling semantics for building deconstruction. In: Proc., 12wt International Conference on Computer Graphics Theory and Applications. 2017,.

[19] Paoluzzi A, DiCarlo A, Furiani F, Jirík M. CAD models from medical images using LAR. Computer-Aided Design and Applications 2016;13(6):747–59. doi:10.1080/16864360.2016.1168216.

[20] Clementi G, Salvati D, Scorzelli G, Paoluzzi A, Pascucci V. Progressive extraction of neural models from high-resolution 3D images of brain. In: 13th Int. Conf. on CAD & Applications. Vancouver, BC, Canada; 2016,.