Computational topology and Boolean operations with LAR (sparse arrays)

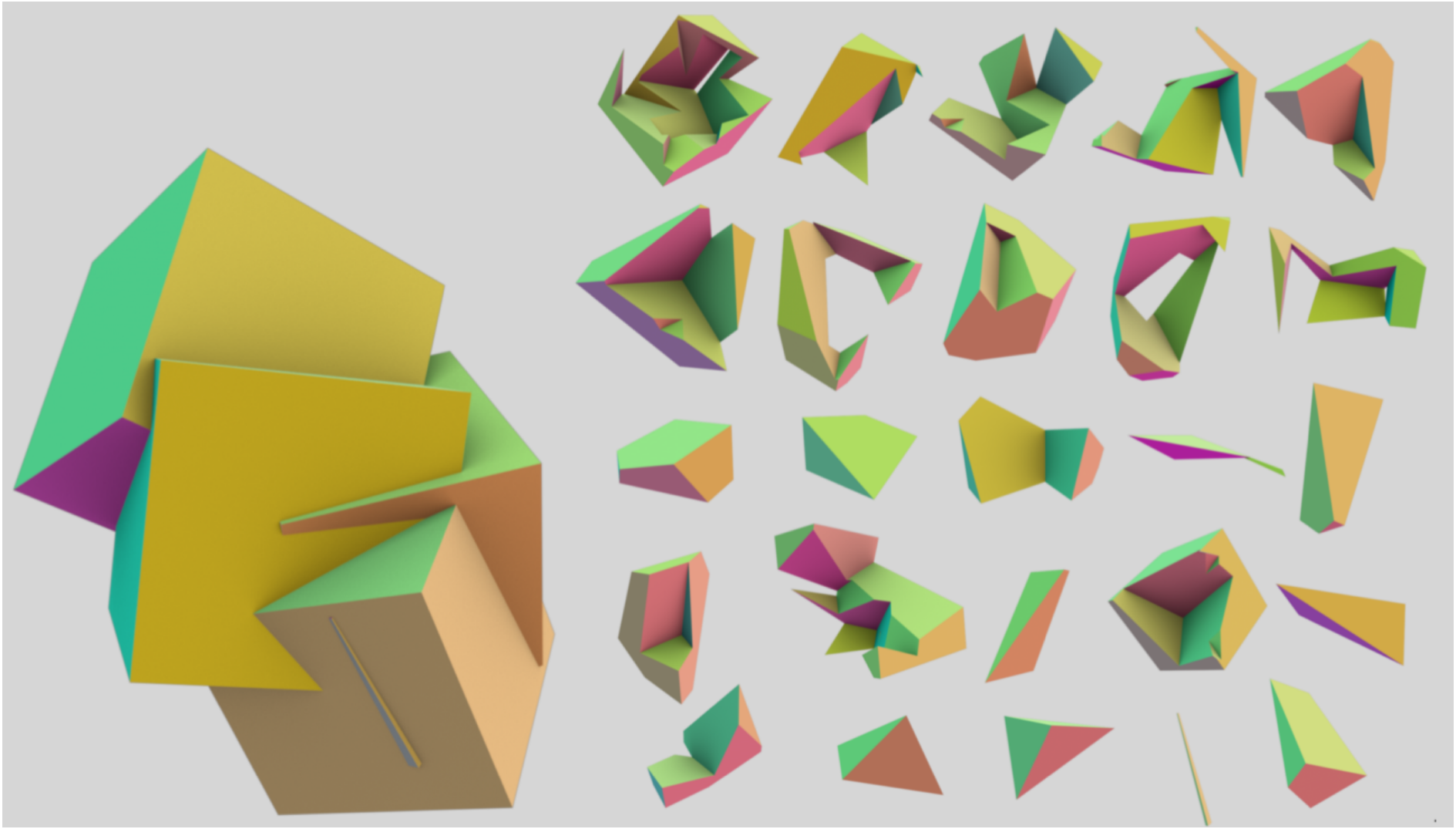
Alberto Paoluzzi

3D-Day — Roma Tor Vergata — October 9, 2019

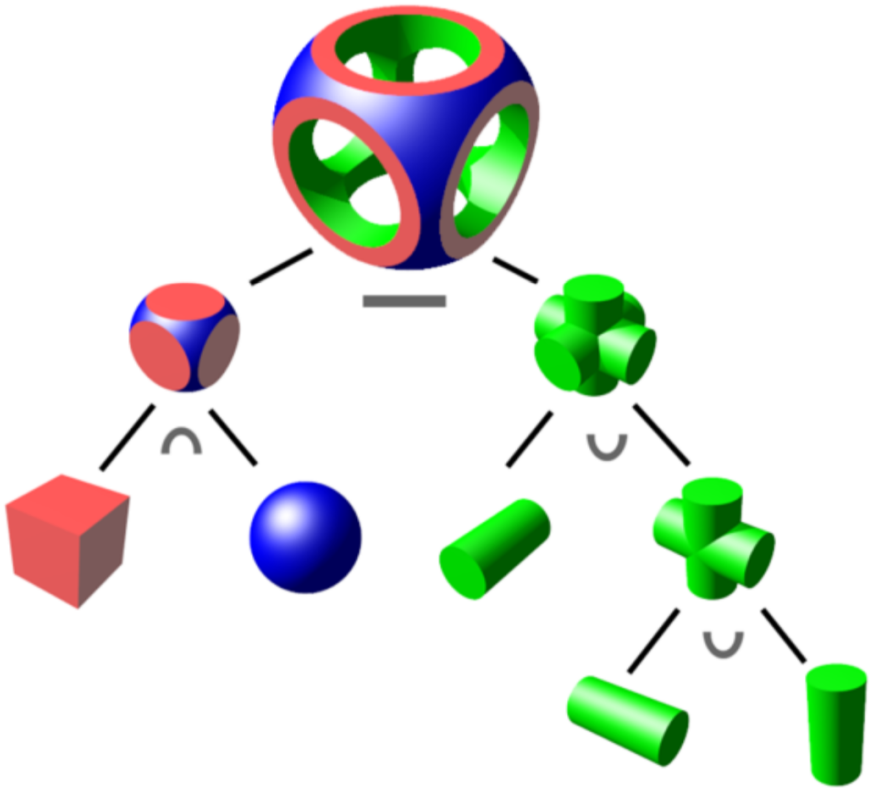
1. Introduction
2. Chain complex of space partition
3. Assessment of Solid Boolean Algebras

Introduction

Space Arrangement and Algebras

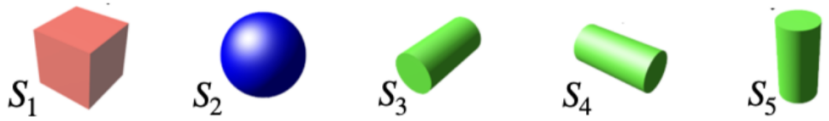


Boolean Algebras of SOLIDS



CSG-tree

$\mathbb{E}^3$ arrangement induced by $\{S_i\}$



$$C_3 \underset{\partial_3}{\overset{\delta^2}{\leftrightarrows}} C_2 \underset{\partial_2}{\overset{\delta^1}{\leftrightarrows}} C_1 \underset{\partial_1}{\overset{\delta^0}{\leftrightarrows}} C_0$$

$$\chi : \{S_1, \ldots, S_5\} \to C_3$$

$$X := [\chi(S_1); \ldots; \chi(S_4); \chi(S_5)]$$

$$X_j = X[:, j]$$

$$((X_1) \cap (X_2)) - ((X_3) \cup ((X_4) \cup (X_5)))$$

binary formula evaluation

Chain complex of space partition

From Arrangement to Chain Complex

Given a collection $\mathcal{S}$ of geometric objects[1], the subject of this paper is computing the topology of their space arrangement $\mathcal{A}(\mathcal{S})$ as a chain complex, i.e.,~as a short exact sequence of linear spaces C_i of (co)chains and linear boundary/coboundary maps ∂_p and $\delta_p = \partial_{p+1}^\top$ between them:

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

[1] Examples include, but are not limited to: line segments, quads, triangles, polygons, meshes, pixels, voxels, volume images, B-reps, etc. In mathematical terms, a geometric object is a topological space embedded in some $\mathbb{E}^d$ [**?**].

Arrangement pipeline

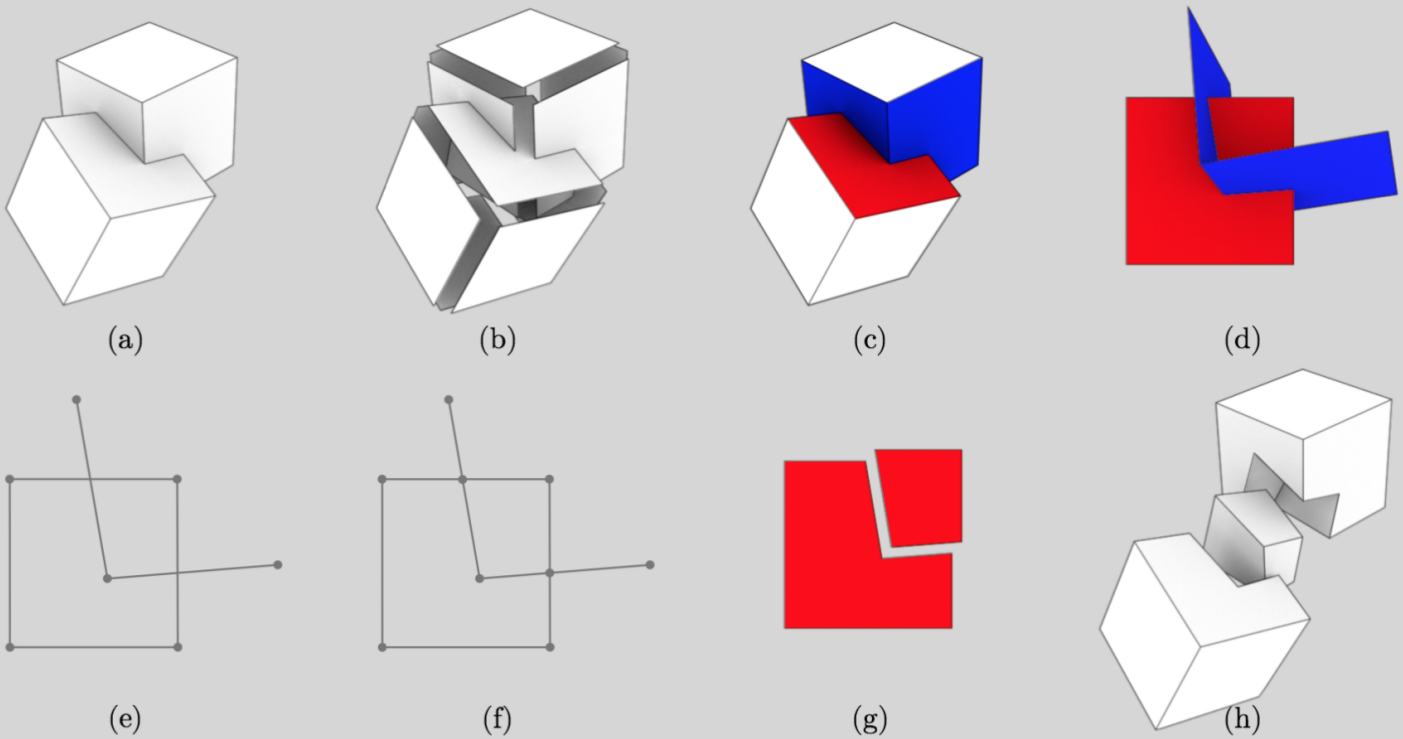


Fig. 2. Cartoon display of the computational pipeline: (a) two solids in $\mathcal{S}$; (b) the exploded input collection $\mathcal{S}_2$ in $\mathbb{E}^3$; (c) 2-cell σ (red) and the set $\Sigma(\sigma)$ (blue) of possible intersection; (d) $\sigma \cup \Sigma$ affinely mapped on $z = 0$; (e) reduction to a set of 1D segments in $\mathbb{E}^2$ via intersection with $z = 0$; (f) pairwise intersections; (g) exploded U_2 basis of C_2 generated as columns of $\partial_2 : C_2 \longrightarrow C_1$; (h) exploded U_3 basis of C_3 generated as columns of operator's $\partial_3 : C_3 \longrightarrow C_2$ sparse matrix, via the TGW algorithm in 3D.

Arrangement pipeline

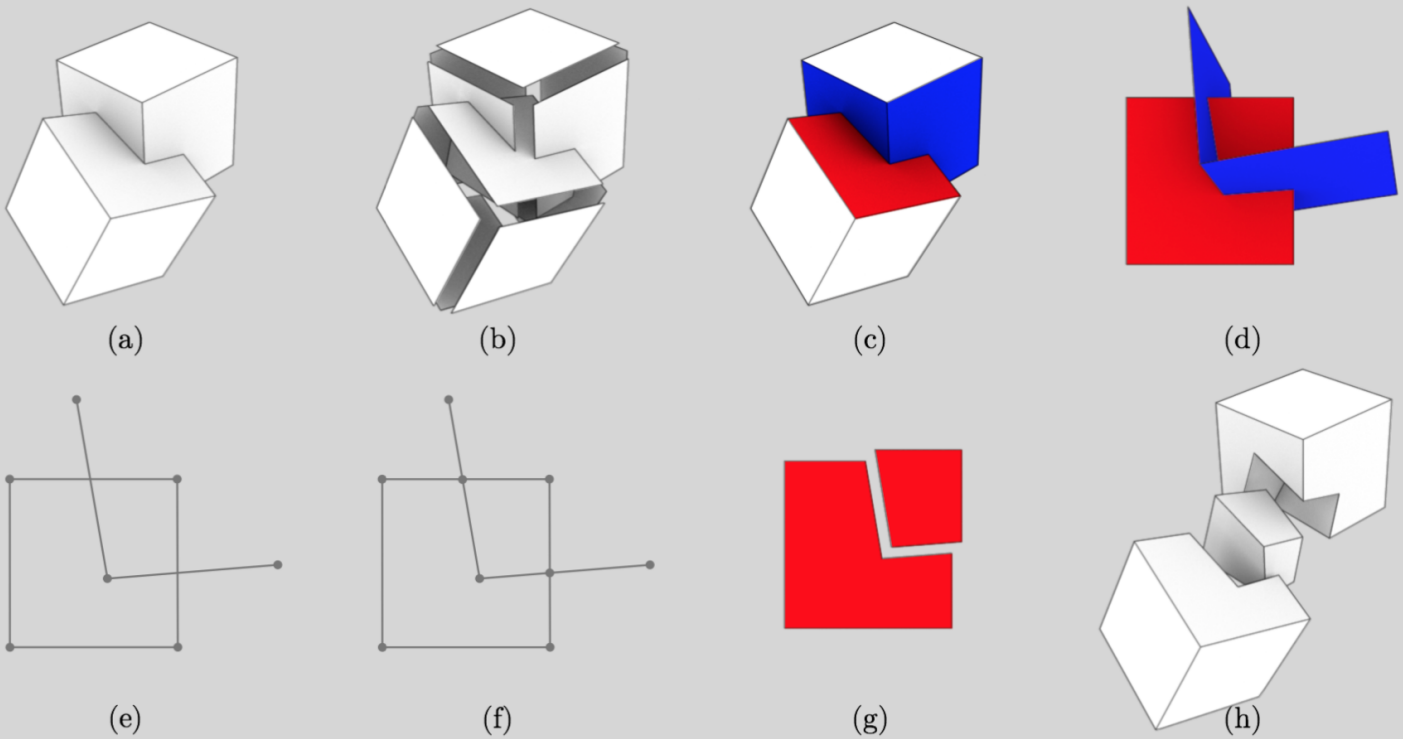
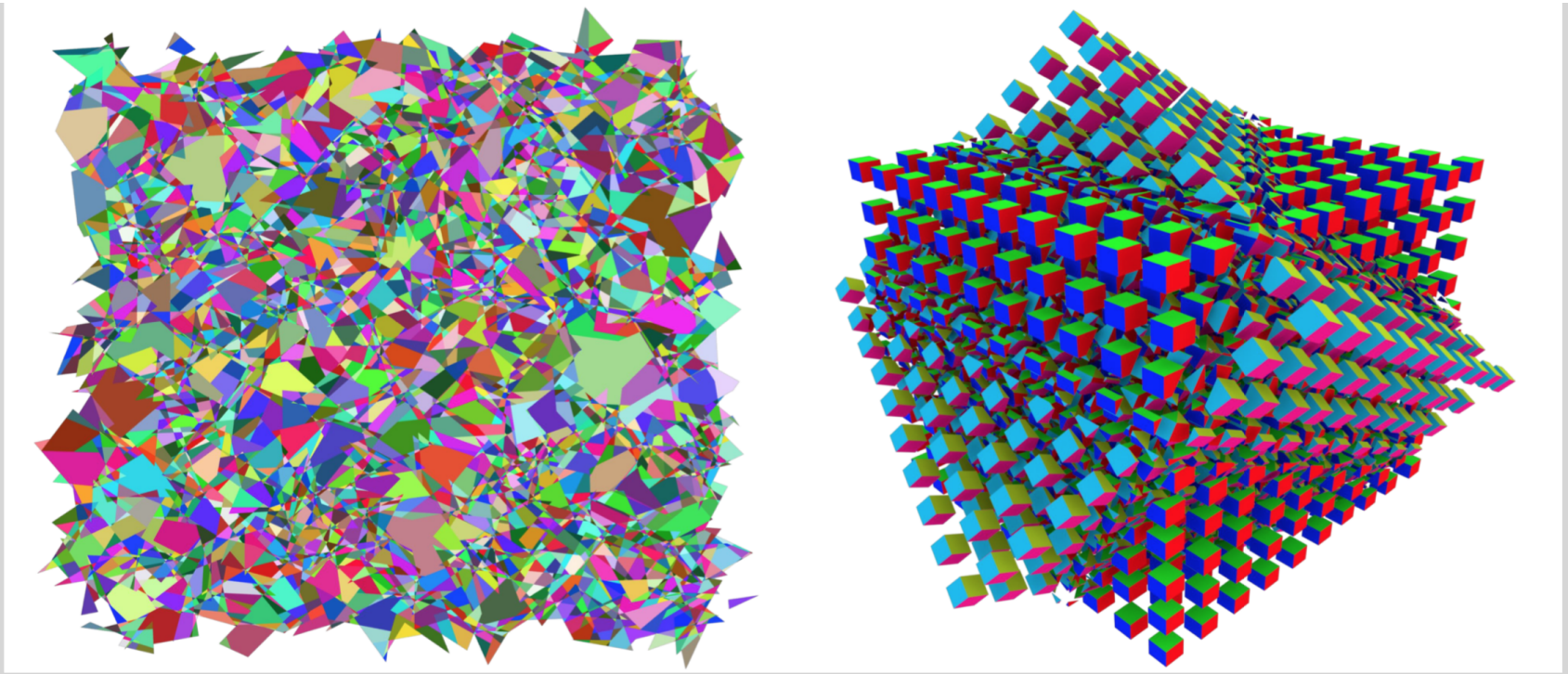


Fig. 2. Cartoon display of the computational pipeline: (a) two solids in $\mathcal{S}$; (b) the exploded input collection $\mathcal{S}_2$ in $\mathbb{E}^3$; (c) 2-cell σ (red) and the set $\Sigma(\sigma)$ (blue) of possible intersection; (d) $\sigma \cup \Sigma$ affinely mapped on $z = 0$; (e) reduction to a set of 1D segments in $\mathbb{E}^2$ via intersection with $z = 0$; (f) pairwise intersections; (g) exploded U_2 basis of C_2 generated as columns of $\partial_2 : C_2 \longrightarrow C_1$; (h) exploded U_3 basis of C_3 generated as columns of operator's $\partial_3 : C_3 \longrightarrow C_2$ sparse matrix, via the TGW algorithm in 3D.

Arrangements, (co)boundary matrices



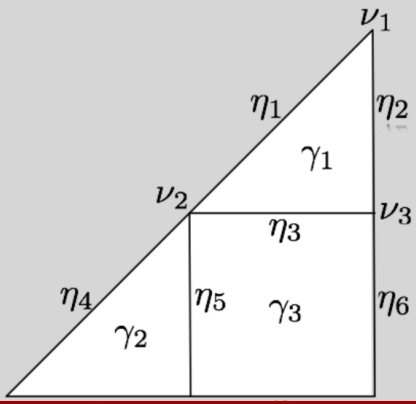
(a) (b)

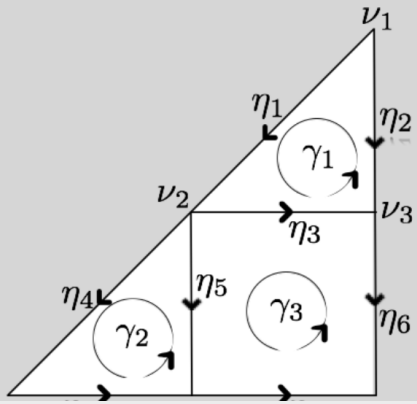
Fig. 1. Examples: (a) Regularized 2D arrangement X_2 of the plane generated by a set of random line segments. The Euler characteristic is $\chi = \chi_0 - \chi_1 + \chi_2 = 11361 - 20813 + 9454 = 2$; (b) merge of two 3-complexes with 2×10^3 3-cells. The Euler characteristic (of the non-exploded resulting 3-complex) is $\chi = \chi_0 - \chi_1 + \chi_2 - \chi_3 = 8787 - 26732 + 26600 - 8655 = 0$. This count includes the outer (unbounded) 2-cell or 3-cell, respectively, that are also computed by theTopological Gift Wrapping algorithm TGW (see Section 2.3). The Euler characteristic of the d-sphere is $\chi = 1 + (-1)^d = 2$ or 0 for either even or odd space dimension d.

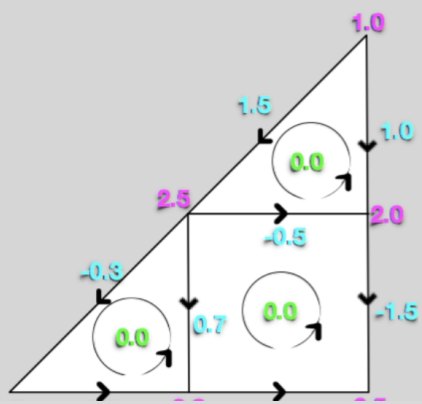
Arrangements, (co)boundary matrices

Example A.5 (Boundary). The boundary operators are maps $C_p \to C_{p-1}$, with $1 \leq p \leq d$. Hence for a 2-complex we have two operators, denoted as $\partial_2 : C_2 \to C_1$ and $\partial_1 : C_1 \to C_0$, respectively. Since they are linear maps between linear spaces, may be represented by matrices of coefficients $[\partial_2]$ and $[\partial_1]$ from the underlying field $\mathbb{F}$. For the unsigned and the signed case (Figures 9a and 9b) we have, respectively:

$$[\partial_2] = \begin{pmatrix} 1 & 0 & 0 \\ 1 & 0 & 0 \\ 1 & 0 & 1 \\ 0 & 1 & 0 \\ 0 & 1 & 1 \\ 0 & 0 & 1 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}, \text{ and } \quad [\partial_2'] = \begin{pmatrix} 1 & 0 & 0 \\ -1 & 0 & 0 \\ 1 & 0 & -1 \\ 0 & 1 & 0 \\ 0 & -1 & 1 \\ 0 & 0 & -1 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}. \tag{2}$$







2-cell partition

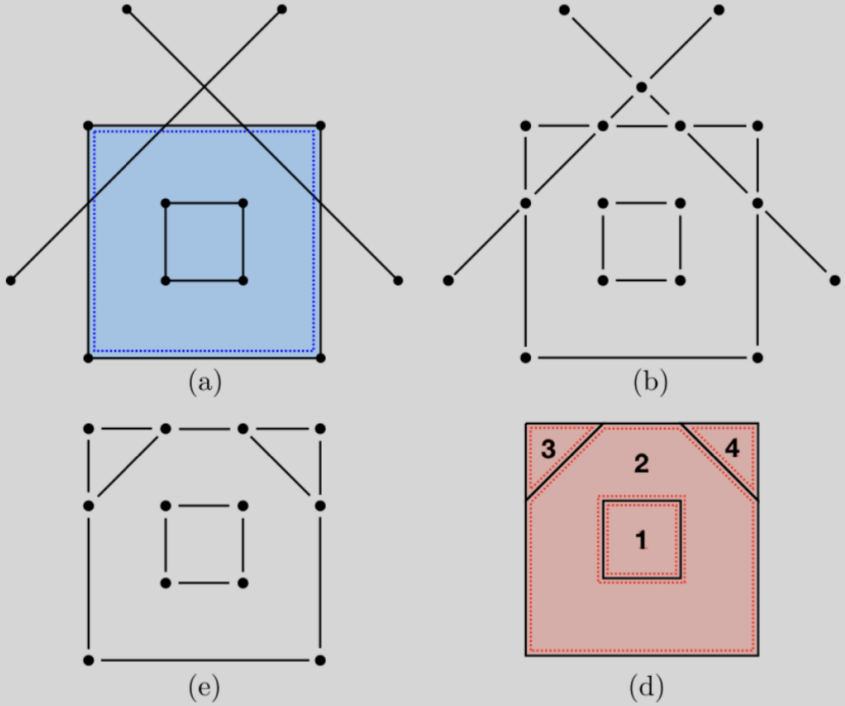


Fig. 3. Basic case: computation of the regularized arrangement of a set of lines in $\mathbb{E}^2$: (a) the input, *i.e.*, the 2-cell σ (signed blue) and the line segment intersections of $\Sigma(\sigma)$ with $z = 0$; (b) all pairwise intersections of 1-cells; (c) removal of the 1-subcomplex external to $\partial\sigma$; (d) the 2-chain generated by $\sigma \cup \Sigma$ via TGW in 2D.

Quotient set computations (Congruence)

$$C_2(U_2/R_2) \xrightarrow{\partial_2} C_1(U_1/R_1) \xrightarrow{\partial_1} C_0(U_0/R_0),$$

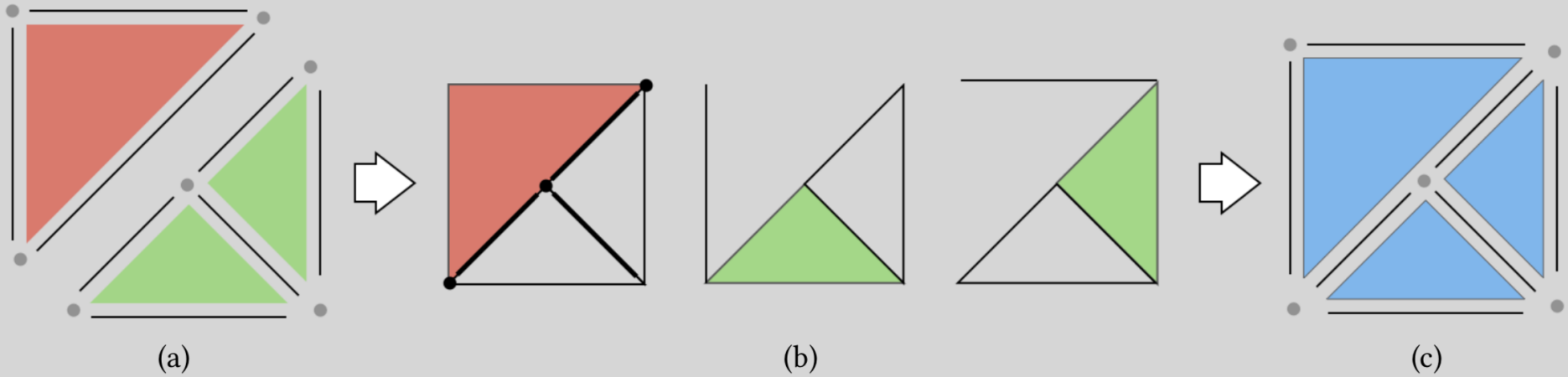
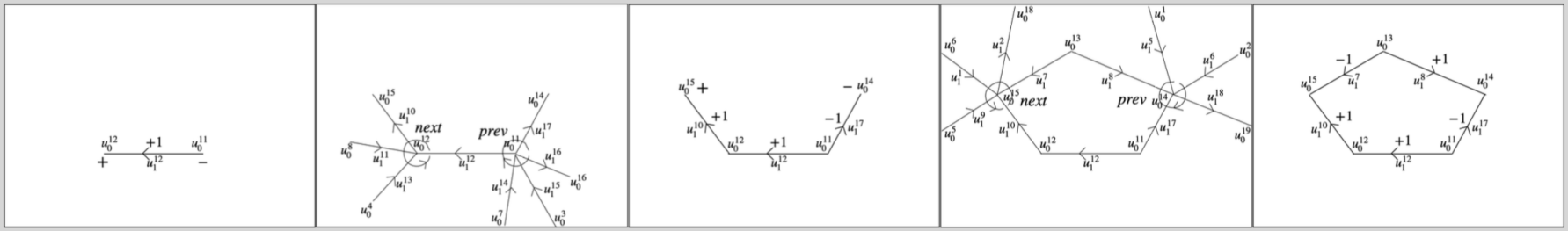


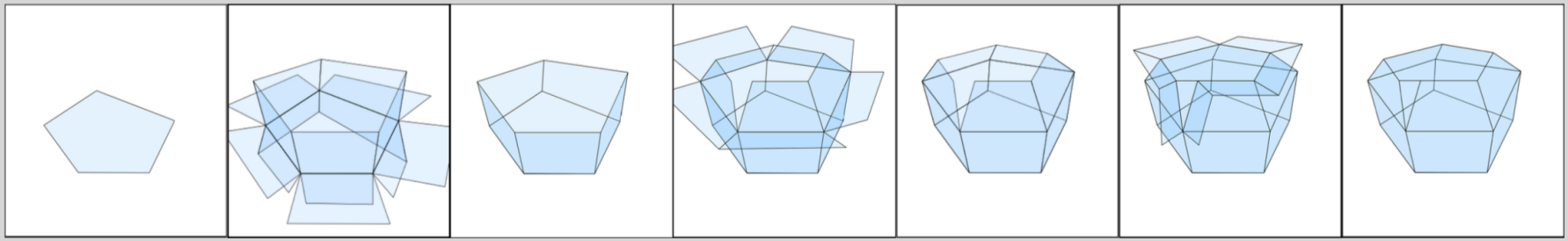
Fig. 4. Merge of two 2D complexes with incompatible boundaries: (a) Input data $\mathcal{S}_2 = \Sigma^1 \cup \Sigma^2$; (b) independent fragmentation of 2-cells $\sigma \in \mathcal{S}_2$ induced by $\mathcal{I}(\sigma)$; (c) *local* arrangement $X_2 = \mathcal{A}(\mathcal{S}_1(\mathcal{S}_2))$ generated by *Merge*, 2D *TGW*, and *Congruence* algorithm pipeline.

3-boundary operator ∂_3



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

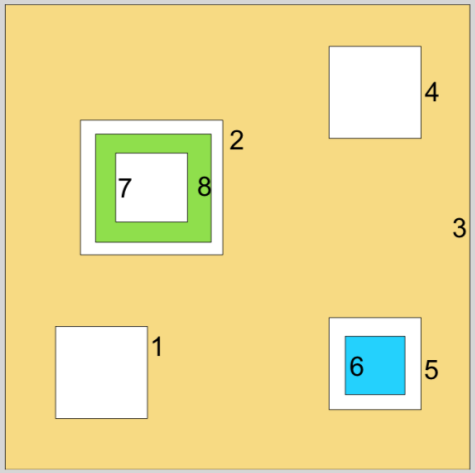
Fig. 5. Extraction of a minimal 1-cycle from $\mathcal{A}(X_1)$: (a) the initial value for $c \in C_1$ and the signs of its oriented boundary; (b) cyclic subgroups on $\delta\partial c$; (c) new (coherently oriented) value of c and ∂c; (d) cyclic subgroups on $\delta\partial c$; (e) final value of c, with $\partial c = \emptyset$. The step-by-step computation is discussed in Example A.1.



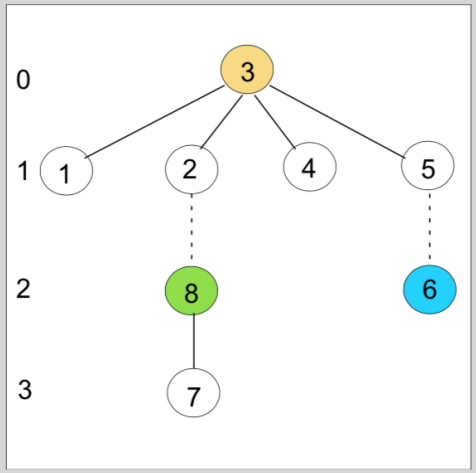
(a) (b) (c) (d) (e) (f) (g)

Fig. 6. Extraction of a minimal 2-cycle from $\mathcal{A}(X_2)$: (a) initial (0-th) value for $c \in C_2$; (b) cyclic subgroups on $\delta\partial c$; (c) 1-st value of c; (d) cyclic subgroups on $\delta\partial c$; (e) 2-nd value of c; (f) cyclic subgroups on $\delta\partial c$; (g) 3-rd value of c, such that $\partial c = 0$, hence stop.

Transitive reduction of shell poset



(a)



(b)

$$R = \begin{pmatrix} - & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & - & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & - & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & - & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & - & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & - & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & - & 1 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & - \end{pmatrix}$$

(c)

Fig. 7. Non intersecting cycles within a 2D cellular complex with three connected components and only three cells, denoted by the image colors: (a) cellular complex; (b) graph of the *reduced* containment relation R between shells, with dashed arcs of even depth index. Removing the dashed arcs produces a *forest* of small *trees*; (c) matrix of transitively reduced R. Note that the ones equate the number of edges in the graph.

Topology computing with chains

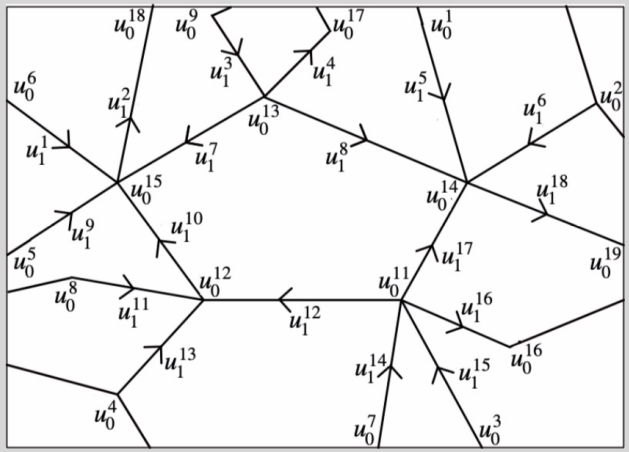
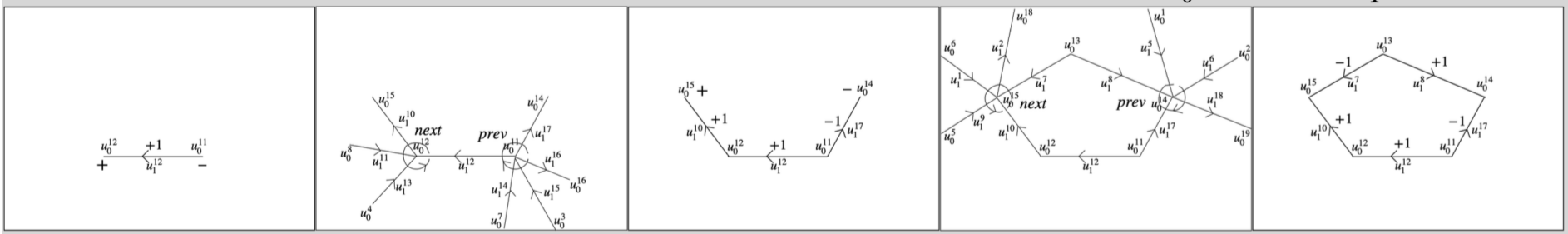


Fig. 8. A portion of the 1-complex used by Example A.1, with unit chains $u_0^h \in C_0$ and $u_1^k \in C_1$.



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

TGW – Topological Gift Wrapping Algorithm

ALGORITHM 2: *Computation of signed* $[\partial_d^+]$ *matrix*

/* **Pre-condition:** d equals the space $\mathbb{E}^d$ dimension, such that $(d-1)$-cells are shared by *two* d-cells */
/* */
Input: $[\partial_{d-1}]$ # Compressed Sparse Column (CSC) signed matrix (a_{ij}), where $a_{ij} \in \{-1, 0, 1\}$
Output: $[\partial_d^+]$ # CSC signed matrix of $(d-1)$-cycles
$[\partial_d^+] = [\,]$; $m, n = [\partial_{d-1}].shape$; $marks = Zeros(n)$ # initializations
while Sum($marks$) $< 2n$ **do**
  $\sigma = Choose(marks)$ # select the $(d-1)$-cell seed of the column extraction
  if $marks[\sigma] == 0$ **then** $[c_{d-1}] = [\sigma]$
  else if $marks[\sigma] == 1$ **then** $[c_{d-1}] = [-\sigma]$
  $[c_{d-2}] = [\partial_{d-1}]\,[c_{d-1}]$ # compute boundary c_{d-2} of seed cell
  while $[c_{d-2}] \neq [\,]$ **do** # loop until boundary becomes empty
    $corolla = [\,]$
    for $\tau \in c_{d-2}$ **do** # for each "hinge" τ cell
      $[b_{d-1}] = [\tau]^t\,[\partial_{d-1}]$ # compute the τ coboundary
      $pivot = \{|b_{d-1}|\} \cap \{|c_{d-1}|\}$ # compute the τ support
      if $\tau > 0$ **then** $adj = Next(pivot, Ord(b_{d-1}))$ # compute the new adj cell
      else if $\tau < 0$ **then** $adj = Prev(pivot, Ord(b_{d-1}))$
      if $\partial_{d-1}[\tau, adj] \neq \partial_{d-1}[\tau, pivot]$ **then** $corolla[adj] = c_{d-1}[pivot]$ # orient adj
      else $corolla[adj] = -(c_{d-1}[pivot])$
    end
    $[c_{d-1}]$ += $corolla$ # insert $corolla$ cells in current c_{d-1}
    $[c_{d-2}] = [\partial_{d-1}]\,[c_{d-1}]$ # compute again the boundary of c_{d-1}
  end
  for $\sigma \in c_{d-1}$ **do** $marks[\sigma]$ += 1 # update the counters of used cells
  $[\partial_d^+]$ += $[c_{d-1}]$ # append a new column to $[\partial_d^+]$
end

The whole picture

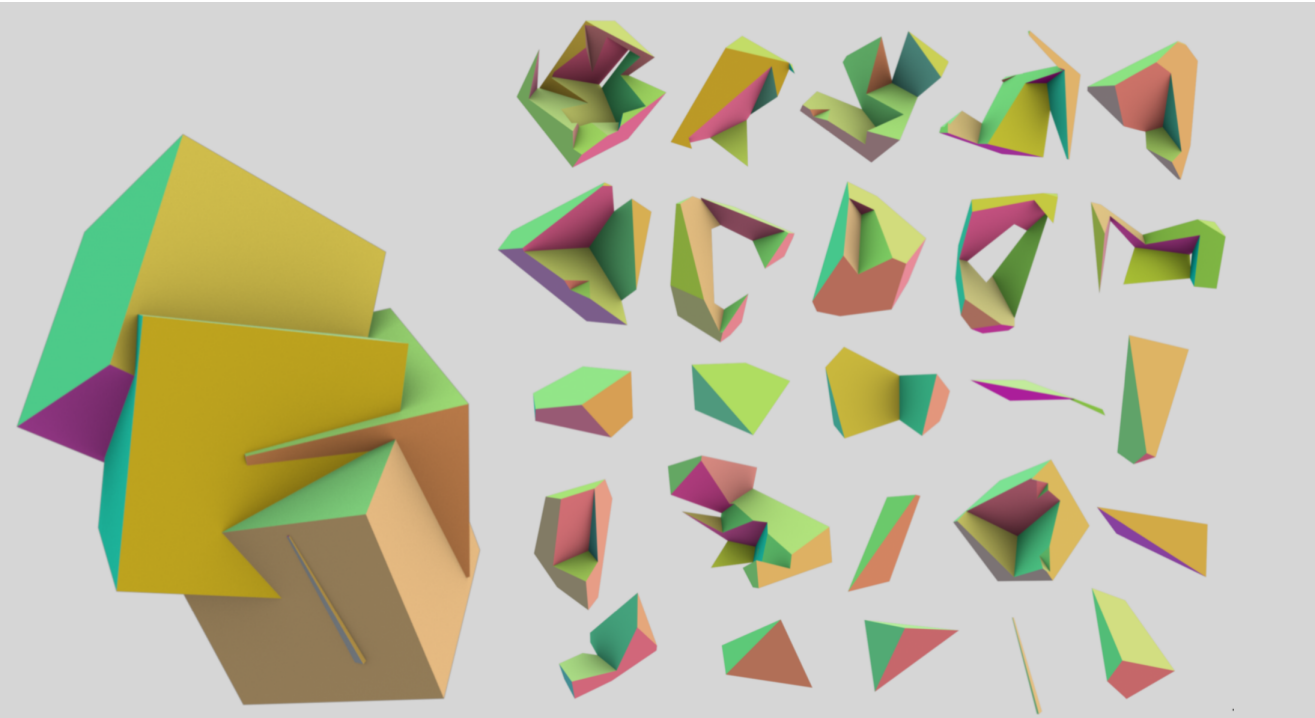
Input Facet selection, *i.e.*, construction of the collection $\mathcal{S}_{d-1}$ from $\mathcal{S}_d$, using LAR.
Indexing Spatial index made by intersection of d interval-trees on bounding boxes of $\sigma \in \mathcal{S}_{d-1}$.
Decomposition Pairwise $z = 0$ intersection of line segments in $\sigma \cup \mathcal{I}(\sigma)$, for each $\sigma \in \mathcal{S}_{d-1}$.
Congruence Graded bases of equivalence classes $C_k(U_k)$, with $U_k = X_k/R_k$ for $0 \leq k \leq 2$.
Connection Extraction of $(X^p_{d-1}, \partial^p_{d-1})$, maximal connected components of X_{d-1} $(0 \leq p \leq h)$.
Bases Computation of redundant cycle basis $[\partial^+_d]^p$ for each p-component, via TGW.
Boundaries Accumulation into H += $[o]^p$ (hole-set) of outer boundary cycle from each $[\partial^+_d]^p$.
Containment Computation of antisymmetric containment relation S between $[o]^p$ holes in H.
Reduction Transitive R reduction of S and generation of forest of flat trees $\langle [o_d]^p, [\partial_d]^p \rangle$.
Adjoining of roots $[o_d]^r$ to (unique) outer cell, and non-roots $[\partial^+_d]^q$ to container cells.
Assembling Quasi-block-diagonal assembly of matrices relatives to isolated components $[\partial_d]^p$.
Output Global boundary map $[\partial_d]$ of $\mathcal{A}(\mathcal{S}_{d-1})$, and reconstruction of 0-chains of d-cells in X_d.

Assessment of Solid Boolean Algebras

Algebra of sets

Finite Boolean Algebras for Solid Geometry using Julia's Sparse Arrays

Alberto Paoluzzi,Vadim Shapiro,Antonio DiCarlo,Giorgio Scorzelli,Elia Onofri



The set of join-irreducible atoms of the Boolean Algebra generated by a partition of $\mathbb{E}^3$, and represented as a basis of 3-chains.

Binary representation of Boolean terms

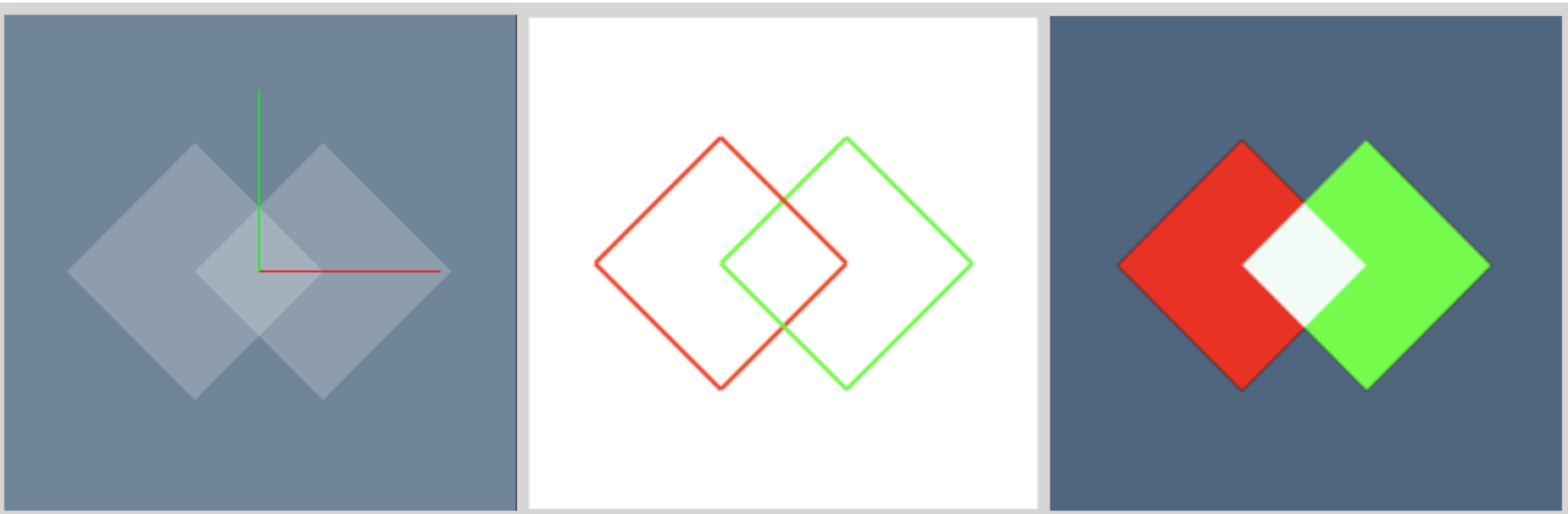


Figure 1: *Space arrangement* generated in $\mathbb{E}^2$ by: (a) two overlapping 2-cells A and B; (b) the 4+4 line segments (1-cells) in their skeletons, generating the *four* 2-cells in (c): blue (c_1), red (c_2), white (c_3), and green (c_4). The first (c_1) is the *outer* or *exterior* cell Ω, which is the complement of the sum of the others.

Binary representation of Boolean terms

Table 1
Truth table associating 2-cells c_i (i=1,...,4) in U_2 (rows) and solid variables A, B and $\Omega = \overline{A \cup B}$ (columns). See Figure 1.

U_2	Ω	A	B
c_1	1	0	0
c_2	0	1	0
c_3	0	1	1
c_4	0	0	1

Table 2
Truth table providing the complete enumeration of elements S of the finite Boolean algebra $\mathcal{A}$ generated by two solid objects A, B and by four atoms c_i in the basis U_2 of chain space C_2 associated to the arrangement $\mathcal{A}(S)$, with $S = \{\partial(A), \partial(B)\}$.

U_2	$X = \mathbb{E}^2$	A	B	$\overline{A \cup B}$	$A \cup B$	$A \cap B$	$A \setminus B$	$B \setminus A$	$A \oplus B$	$\overline{A \setminus B}$	$\overline{B}$	$\overline{B \setminus A}$	$\overline{A}$	$\overline{A \oplus B}$	$\overline{A \cap B}$	$\emptyset$
c_1	1	0	0	1	0	0	0	0	0	1	1	1	1	1	1	0
c_2	1	1	0	0	1	0	1	0	1	0	1	1	0	0	1	0
c_3	1	1	1	0	1	1	0	0	0	1	0	1	0	1	0	0
c_4	1	0	1	0	1	0	0	1	1	1	0	0	1	0	1	0

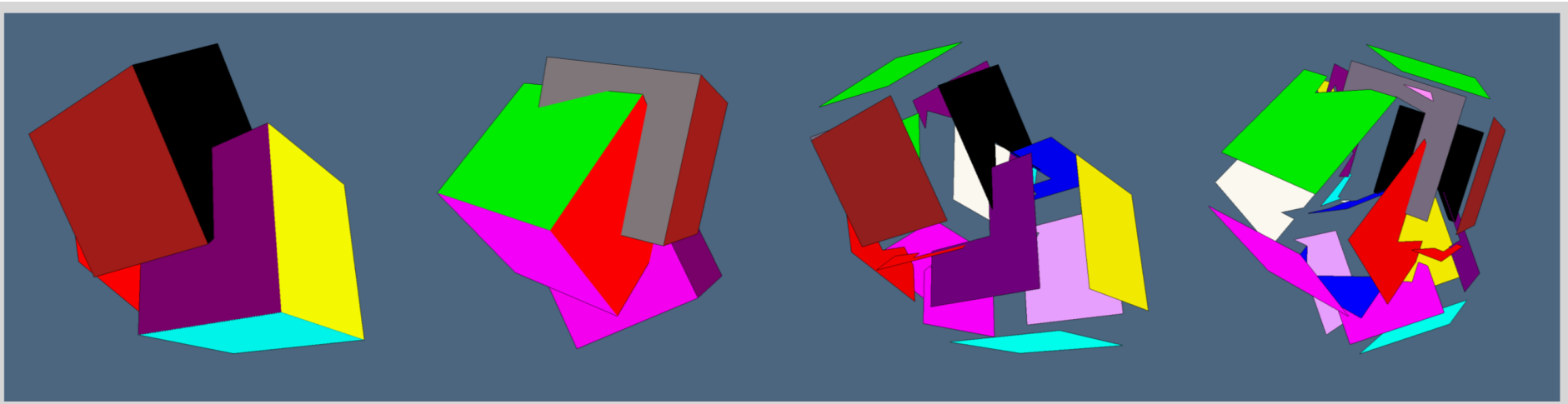
Generation of space arrangement of CSG terms

Example 5. Space arrangement from `assembly` tree. *Let us consider the* `assembly` *constructed by putting together three instances of a unit cube, properly rotated and translated. The semantics of* `Lar.Struct()` *is similar to that of PHIGS structures (Kasper and Arns, 1993; Paoluzzi, 2003):*

```
julia> m,n,p = 1,1,1
    Lar = LinearAlgebraicRepresentation;
    V,(VV,EV,FV,CV) = Lar.cuboidGrid([m,n,p],true);
    cube = V,FV,EV;
julia> assembly = Lar.Struct([ cube,
    Lar.t(.3,.4,.25), Lar.r(pi/5,0,0), Lar.r(0,0,pi/12), cube,
    Lar.t(-.2,.4,-.2), Lar.r(0,pi/5,0), Lar.r(0,pi/12,0), cube ]);
julia> W, EV, FE, CF, boolmatrix = Lar.bool3d(assembly);
```

The application of function `Lar.bool3d()` *to* `assembly` *returns the* `(geometry,topology)` *of 3D space partition generated by it. Here* `geometry` *is given by the embedding matrix* `W` *of vertices (0-cells), and* `topology` *is given by the three sparse matrices* `CF, FE, EV`, i.e., $\delta_3, \delta_2, \delta_1$, *of chain complex describing the* $\mathcal{A}$(`assembly`) *arrangement.*

Generation of space arrangement of CSG terms



(a) (b) (c) (d) .

Figure 6: Boolean union $A \cap B \cap C$ of three cubes (from `assembly` of Example 5), with 2-cells in different colors: (a) view from the front; (b) view from the back; (c) front with exploded 2-cells; (d) back with exploded 2-cells. Only the boundary 2-cells are displayed. The space partition generated other 2-cells in the interior.

Generation of space arrangement of CSG terms

Example 6. Boolean matrix. *The array value of type* `Bool` *returned within the variable* `boolmatrix` *contains by column the results of efficient point-solid containment tests (SMC) for atomic 3-cells (rows), w.r.t. the terms of* $\mathbb{E}^3$ *partition: the outer 3-cell* Ω *and each* `cube` *(columns). Of course, the first row and column are trivially computed.*

```
julia> Matrix(boolmatrix)
8x4 Array{Bool,2}:
 true  false  false  false
false  false  false   true
false   true   true  false
false   true   true   true
false   true  false  false
false  false   true  false
false   true  false   true
false  false   true   true
```

Our variables Ω, A, B, C *are extracted from* `boolmatrix` *columns, so describing how each one is partitioned by ordered 3-cells in* U_3*. The whole space is* $X = \Omega \cup A \cup B \cup C$*:*

```
julia> A,B,C = boolmatrix[:,2],boolmatrix[:,3],boolmatrix[:,4]
(Bool[false, false, true, true, true, false, true, false],
 Bool[false, false, true, true, false, true, false, true],
 Bool[false, true, false, true, false, false, true, true])
```

Prefix and Variadic Union of solid terms

6.1. Variadic union

In order to compute the `union` of three affinely transformed instances of the unit `cube`, we consider the `assembly` expression given in Example 5. First we get the $\mathbb{E}^3$ space partition generated by `assembly` through the function `Lar.bool3d`; then we combine the logical arrays `A`, `B`, and `C`, building the value of `BitArray` type for the `union` variable, that stores the logical representation of the specific 3-chain. Let us remark that the bitwise `or` operator (`|`) is applied vectorized to arrays, by inserting a dot character:

```
julia> W, (EV, FE, CF), boolmatrix = Lar.bool3d(assembly);
julia> A,B,C = boolmatrix[:,2],boolmatrix[:,3],boolmatrix[:,4]
julia> union = .|(A, B, C);
julia> @show union;
union = Bool[false, true, true, true, true, true, true, true]
```

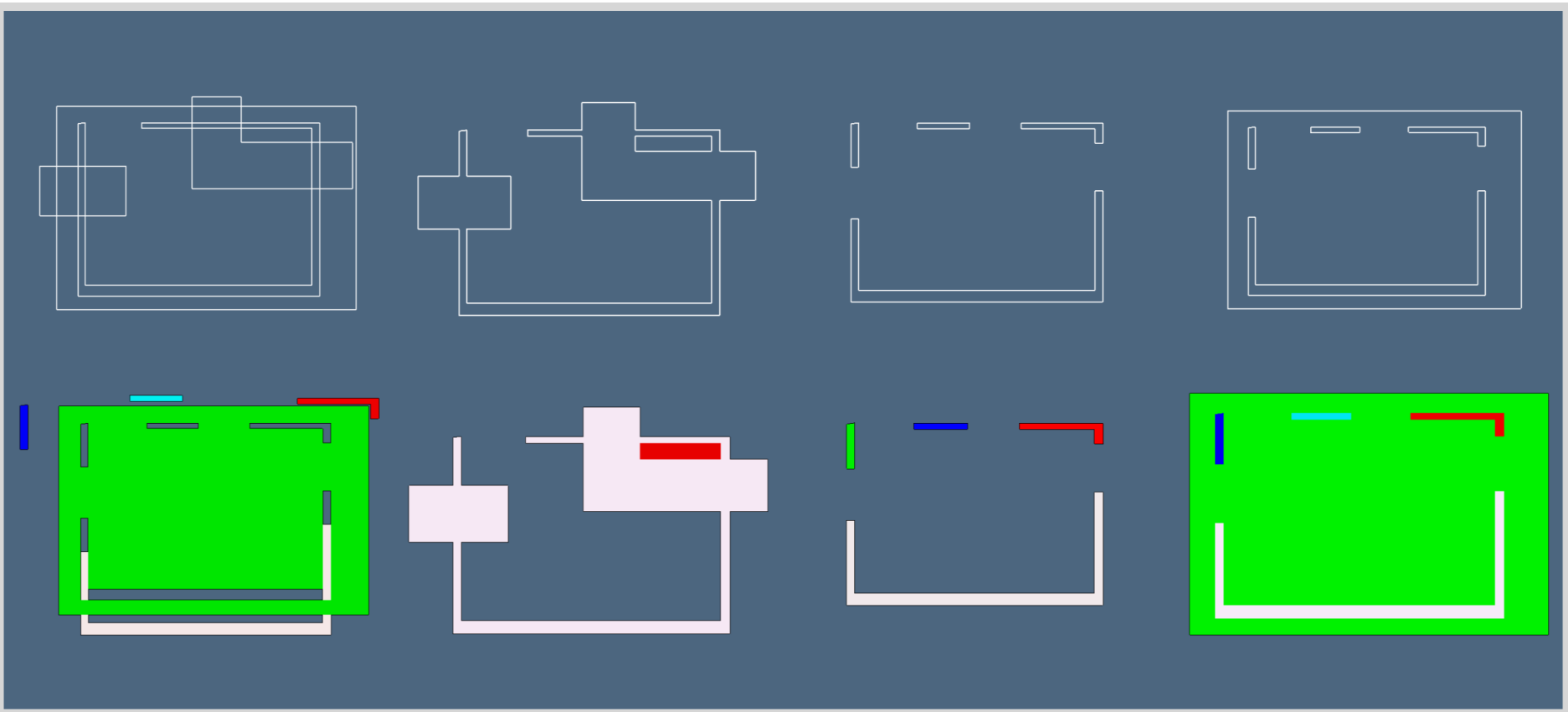
Finally, the boundary 2-cycle `faces` is generated by multiplication of the sparse matrix $[\partial_3]$ (*i.e.*, `CF'`) times the binary translated `union`. The mapping `Bool` $\rightarrow \{0, 1\}$ is applied via a vectorized application of the `Int8` constructor:

```
julia> faces = CF' * Int8.(union);
julia> @show faces;
faces = [1, 0, -1, 0, 0, 0, -1, 0, 1, 0, 1, 0, 0, 0, 1, -1, 0, -1, 0, 0, 1, -1, 0, -1, 1,
0, 0, 0, 1, 1, 0, -1, 0, 0, 1, 0, -1, 0, 0, 0, -1, 1, 0, 1, 0, 0, -1]
```

With $f_k \in U_2$, where U_2 is the basis of chain space C_2 generated by $\mathcal{A}$(`assembly`), we may write in chain notation:

$$\begin{aligned} \texttt{faces} \mapsto f_{A \cup B \cup C} = {} & f_1 - f_3 - f_7 + f_9 + f_{11} + f_{15} - f_{16} - f_{18} + f_{21} - f_{22} - f_{24} \\ & + f_{25} + f_{29} + f_{30} - f_{32} + f_{35} - f_{37} - f_{41} + f_{42} + f_{44} - f_{47} \end{aligned} \tag{1}$$

Example 2D



(a,b) (c,d) (e,f) (h,g) .

Figure 5: Some examples of variadic Booleans, obtained by applying the Boolean operators "`.|`", "`.-`", "`.!`", and "`.&`" to the `assembly` variable, with 1-cells (B-reps) imported from `SVG` files. (a,b) start-to-end: from polygon input via `SVG` to (exploded) difference of outer box and interior walls; (c,d) boundary of union and union of some shapes; (e,f) difference 2-complex and its boundary; (g,h) boundary of outer box minus the previous 2-complex, and the corresponding 2-complex.

Example 3D: ternary difference of cubes

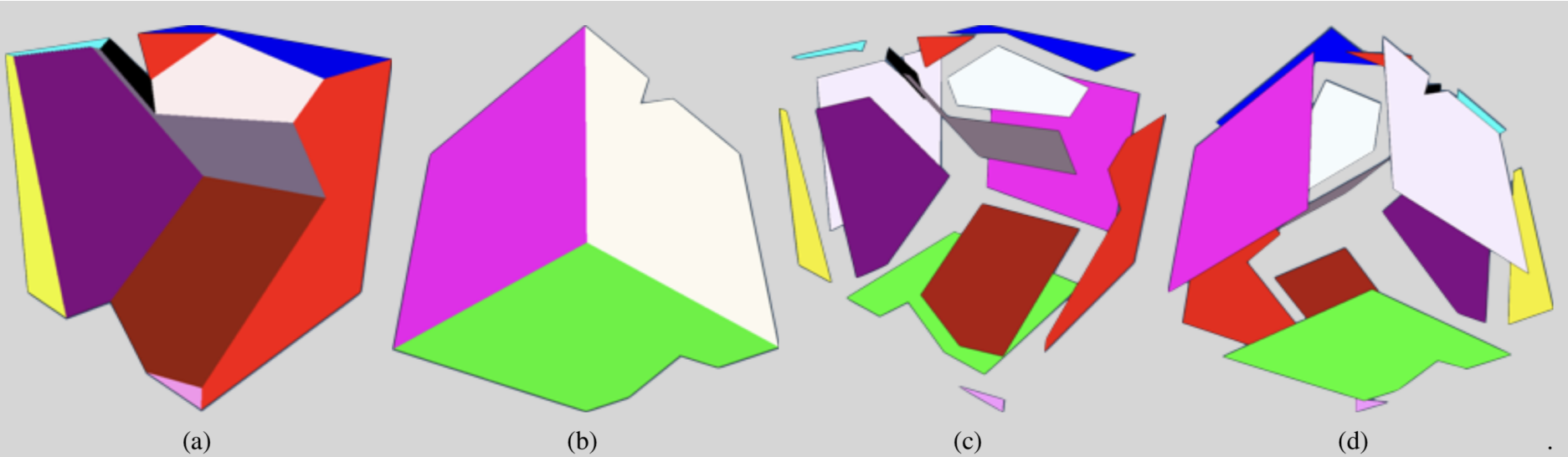


Figure 4: Boolean difference $A \setminus B \setminus C$ of three cubes, with 2-cells in different colors: (a) view from the front; (b) view from the back; (c) front with exploded 2-cells; (d) back with exploded 2-cells. Note that 2-cells of result's boundary may be non-convex.

Summary

1. Basic tools of linear algebra and algebraic topology are used, namely, (sparse) matrices of linear operators and matrix multiplication or filtering. Interval-trees and *kd*-trees introduced only for acceleration.
2. Accuracy of topological computations is guaranteed, since operator matrices satisfy by construction the (graded) constraints $\partial^2 = 0$ and $\delta^2 = 0$.
3. The evaluation of CSG expressions of arbitrary complexity is done in a novel way. All input B-reps are accumulated in a single collection of 2-cells, each of which is operated independently, so generating a collection of local topologies, that are merged by boundary congruence.
4. Once the global space partition is generated, and 3-cells are classified w.r.t. all component solids, via a single point-set containment test, all CSG expressions—of any complexity—can be evaluated simply by bitwise vectorized logical operations.
5. All distinction is removedbetween manifold and non-manifold representations, allowing for mixing B-reps, cellular decompositions of elementary solids, and/or regular grids. This allows for the generation of internal structures, that make stronger andmore resilient the objects to be produced via 3D printing.
6. This approach can be extended to general dimensions and/or implemented on highly parallel computational engines, also using standard computing kernels.