Simploidal Sets

Pascal Lienhardt, Samuel Peltier

Simploidal Sets

Pascal Lienhardt[a], Samuel Peltier[a]

[a] *XLIM, UMR CNRS 7252, Université de Poitiers, France*

Abstract

Simplicial sets and cubical sets are combinatorial structures studied since a long time in Algebraic Topology, and they are used for many applications in Computational Topology, Geometric Modeling, CAD, Computer Graphics, etc.

Simploidal sets are defined in this paper in order to homogenize and generalize simplicial and cubical sets. Basic operations are also defined here, which extend basic simplicial and cubical operations (cone, identification, cartesian product). In order to associate shapes with structures, structural relations between simploidal sets and Bézier spaces are provided.

It is then possible to (simultaneously) handle through a single formalism: simplices, cubes, and more generally cells corresponding to products of simplices, with a very low additional cost, regarding space (data structure), time (construction or computation operations) or software development.

Keywords: Simplicial Sets, Cubical Sets, Simploidal Sets, Bézier Spaces

1. Introduction

N-dimensional simplicial "objects" (or triangulations) are studied since a long time in Algebraic Topology. *Combinatorial structures*, as abstract simplicial complexes or simplicial sets, describe the adjacency and incidence relations of abstract simplices, and a *geometric realization* is associated with any structure; for instance, the geometric realization of an abstract simplicial complex (resp. a simplicial set) is a simplicial complex (resp. a CW-complex) [1, 2] (see also [3, 4]). Such structures (or derived data structures) are used for different purposes, in Computational Topology (e.g. computation of topological properties) [5, 6, 7], Geometric Modeling, CAD and Computer Graphics (e.g. representation of subdivided geometric objects, meshes) [8, 9], etc. In particular for these last applicative fields, structural relations between some simplicial structures and triangular Bézier spaces have been established [10].

N-dimensional cubical "objects" have been studied more recently [11, 12] for similar purposes in Topology, Geometric Modeling, etc. The question of a "best" structure, according to the complexity of the representation, to the definition of construction operations and to their complexities, to the complexity of the computation of topological properties, has been discussed, without clear conclusion [13, 14, 15]. Note that it is always possible to decompose a simplicial (resp. cubical) structure into a cubical (resp. simplicial) structure, but this operation increases the number of cells (simplices or cubes). For instance, if a simplicial structure is decomposed into a cubical structure, which is itself decomposed into a simplicial structure, the resulting structure contains many more simplices than the original simplicial structure (cf. Figure 1 (a, b, c)): it is thus not possible to solve the question of a "best" structure by this conversion process. So, different softwares are developed for similar purposes, based upon simplicial structures or cubical structures.

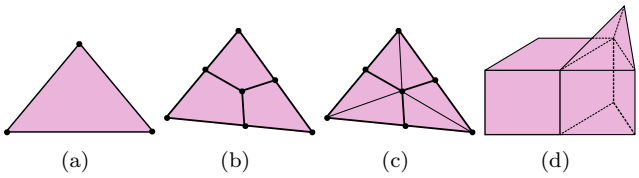

Figure 1: A triangle (a) decomposed into squares (b). The decomposition of (b) into triangles (c) contains more simplices than the original object (a). (d) a tetrahedron and a cube connected by a prism.

A new data structure, *simploidal sets*, is defined in this paper, in order to *homogenize* and to *generalize* simplicial and cubical structures. As a cube is the cartesian product of edges (i.e. 1-dimensional simplices), a simploid is the cartesian product of (any) simplices. A simplex, or a cube, is thus simply a particular simploid.

So, simploidal sets make it possible to :

- handle both simplicial sets or cubical sets ; in other words, simplicial sets or cubical sets can be implemented through simploidal sets ;
- simultaneously handle simplices and cubes, maybe connected through more complex simploids, e.g.

Email addresses: pascal.lienhardt@univ-poitiers.fr (Pascal Lienhardt), samuel.peltier@univ-poitiers.fr (Samuel Peltier)

prisms (cf. Figure 1(d)). This can be interesting for several applications, e.g. related to meshes ;

- handle cellular objects, in which cells are cartesian products of simplices. So, simploidal sets are intrinsically interesting for some applications, for instance for handling assemblies of simploidal Bézier spaces (cf. [16]). For this purpose, the structural relations between simploidal sets and simploidal Bézier spaces are also defined in this paper. They are based upon the structural relations between simplicial sets and triangular Bézier spaces, which are established in section 3 (as far as we know, these relations have not been previously published).

In this paper, basic operations for constructing simploidal sets are also defined, by generalizing basic operations defined for handling simplicial sets (cone, identification, "simplicial" cartesian product) and cubical sets (extrusion, identification).

Since the definition of simploidal sets is very close to that of simplicial and cubical sets (cf. Section 4), the additional cost is very low, regarding space (cost of data structure), time (cost of operations) or software development.

Simploidal sets homogenize and generalize simplicial and cubical sets, which are the most general simplicial and cubical structures. As optimized data structures can be defined for representing subclasses of simplicial (resp. cubical) sets (e.g. abstract simplicial complexes, semi-simplicial sets), it is possible to optimize simploidal sets in order to handle subclasses: cf. for instance semi-simploidal sets ([17]) in section 2.3. Note that in [17], "*simploidal* sets" means "*semi-simploidal*" sets: so, the results presented in this paper extend those of [17]: the main interests of simploidal sets as defined here, related to semi-simploidal sets, are:

- all simplicial and cubical construction operations can be generalized within the simploidal framework, with a very low additional cost;
- accordingly, more types of cells can be handled, improving the versatility of the data structure.

More precisely, semi-simplicial sets are sets of abstract simplices together with *face operators*, which associate with an i-dimensional simplex the $(i-1)$-simplices of its boundary. Any semi-simplicial set can be constructed by two basic operations, the cone and the identification. Semi-cubical sets can be presented in a similar way, but the cells are cubes and the basic operations are the extrusion (i.e. the cartesian product by an edge) and the identification. A first work for generalizing these structures has lead to the definition of semi-simpoidal sets[1] [17], and relations between semi-simploidal sets and products of triangular Bézier spaces have also been established. Simplicial sets extend semi-simplicial sets by adding *degeneracy operators*: this makes it possible for instance to define the cartesian product of simplicial sets. Cubical sets extend semi-cubical sets in the same way. Simploidal sets defined in this paper generalize both simplicial and cubical sets by adding also degeneracy operators to the definition of semi-simploidal sets. For instance, this makes it possible to define the cone operation for simpoidal sets, and thus all basic simplicial and cubical operations can be defined within the simploidal framework.

In 1957, Gugenheim defined *supercomplexes* in order to generalize also simplicial and cubical sets, for the study of topological properties [18]. Even if the basic ideas behind the definitions of simploidal sets and supercomplexes are very close, they are expressed differently. The definition of Gugenheim is very simple, but some aspects remain unclear, for instance when one intend to define construction operations (note that this was not a goal for Gugenheim). The definition of simploidal sets proposed here is more precise, and can be directly implemented; other notions studied here, as relations with Bézier spaces and the definitions of basic operations, are original. For more precisions, see the comparison between simploidal sets and supercomplexes in section 4.5.

In order to understand the definition of simploidal sets, their relations with simploidal Bézier spaces and the definitions of basic operations, all necessary notions are presented in the paper in a progressive way:

- In section 2, the definitions of semi-simplicial, semi-cubical and semi-simploidal sets are recalled, together with the relations with Bézier spaces and the definitions of basic operations;
- In section 3, the definition of simplicial sets are recalled; the precise relations with triangular Bézier spaces are established, and we recall the definition of the cartesian product of simplicial sets, defined in a very simple way thanks to the degeneracy operators;
- simploidal sets are then defined in section 4; we establish the relations with simploidal Bézier spaces, we show in particular that the cone operation can be defined in a simple way thanks to the degeneracy operators. We propose a data structure, basic operations and a comparison with Gugenheim's supercomplexes.

2. Structures Without Degeneracy

This section is mainly based upon the papers [19, 17]. Note that algorithms implementing basic operations for handling semi-simplicial sets and semi-simploidal sets are described in these papers.

2.1. Semi-Simplicial Sets

Combinatorial structure Most well-known triangulated objects are simplicial complexes [1], which are sets

[1] They are called simploidal sets in the original paper [17], but this terminology is modified here in order to be coherent with terminologies related to simplicial and cubical structures.

of geometric simplices satisfying some properties, a i-dimensional simplex (or i-simplex) being the convex hull of $i+1$ linearly independent points. The corresponding combinatorial structure is also well-known: an abstract simplicial complex is a set of abstract simplices, and an abstract i-simplex is a set of $i+1$ (abstract) vertices.

Semi-simplicial sets, or Delta-sets [20, 21] generalize abstract simplicial complexes. A semi-simplicial set is a set of (abstract) simplices on which face operators are defined: more precisely, $i+1$ face operators are defined for a i-simplex σ, associating with σ the (at most) $i+1$ $(i-1)$-simplices of its boundary. Intuitively, if σ is assimilated with a sequence $(v_0, \cdots, v_j, \cdots, v_i)$ of $i+1$ vertices, the image of σ by the j^{th} face operator corresponds to the sequence $(v_0, \cdots, v_{j-1}, v_{j+1}, \cdots, v_i)$. Semi-simplicial sets are more versatile than abstract simplicial complexes, since:

- the definition of a simplex boundary through face operators makes it possible to handle i-simplices which are incident to less than $(i+1)$ vertices (cf. figure 2);
- simplices are not necessarily linearly embedded; the relations which make it possible to directly associate them with triangular Bézier simplices are recalled below.

Definition 1. *(cf. Figure 2) A n-dimensional semi-simplicial set $S = (K, (d_j))$ is a family of sets $K = (K^i)_{i\in[0..n]}$ together with face operators: $d_j : K^i \to K^{i-1}$ for $1 \leq i \leq n$ and $0 \leq j \leq i$, which satisfy: $\forall i, 2 \leq i \leq n, \forall j, l, 0 \leq l < j \leq i, \forall \sigma \in K^i, \sigma d_j d_l = \sigma d_l d_{j-1}$.*

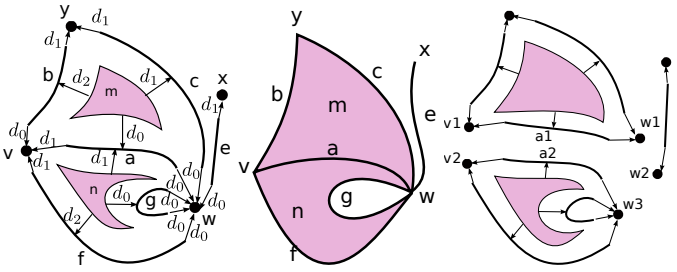

Figure 2: *a*) A semi-simplicial set, and *b*) a geometric representation. The semi-simplicial set of *a*) can be obtained from *c*) by identifying the vertices $v1$ and $v2$, the vertices $w1$, $w2$ and $w3$, and the edges $a1$ and $a2$.

An element σ of K^i is a i-simplex, and σd_j is its j^{th}-face[2]. Simplex τ is a *face* of σ if it can be obtained from σ by applying a non-empty sequence of face operators. The *boundary* of σ is the subset of S restricted to the faces of σ. The *star* of σ is the set of simplices from which σ is a face. σ is a *main* simplex if its star is empty. The boundary of a *complete* i-simplex contains exactly $i+1$ 0-simplices. For example in figure $2(a, b)$: y and b are faces of m; the boundary of n is the set $\{a, g, f, v, w\}$; the star of w is the set $\{f, g, a, c, e, n, m\}$; m,n and e are the main simplices; all simplices but n and g are complete. This terminology can be extended in a straightforward way for all structures described in this paper.

[2] σd_j denotes $d_j(\sigma)$.

Bézier Embedding As abstract simplicial complexes can be associated with simplicial complexes, semi-simplicial sets can be associated with triangular Bézier spaces. A triangular Bézier space is a set of Bézier simplices satisfying some conditions. More precisely, the set of i-dimensional multi-indices Γ^i_d of degree d is defined by $\Gamma^i_d = \{\alpha = (\alpha_0, \cdots, \alpha_i) \in \mathbb{N}^{i+1} \mid |\alpha| = \alpha_0 + \cdots + \alpha_i = d\}$. Multi-index $(\alpha_0, \cdots, \alpha_i)$ will be denoted $\alpha_0 \cdots \alpha_i$: for instance, $\Gamma^1_3 = \{03, 12, 21, 30\}$, and $\Gamma^2_3 = \{003, 012, 021, 030, 102, 111, 120, 201, 210, 300\}$. Multivariate Bernstein polynomials of degree d are defined by $B^d_\alpha(u) = \binom{d}{\alpha} u_0^{\alpha_0} \cdots u_i^{\alpha_i}$, with $\alpha \in \Gamma^i_d$, $\binom{d}{\alpha} = \frac{d!}{\alpha_0! \cdots \alpha_i!}$ are multinomial coefficients. $\{u_j\}$ are the barycentric coordinates of u, a point of the standard i-simplex, i.e. $\{u_j\}$ satisfies: $\forall j, 0 \leq j \leq i, 0 \leq u_j \leq 1$ et $\sum_{j=0}^i u_j = 1$.

Definition 2. *(cf. Figure 3) A i-dimensional Bézier simplex of degree d is defined by $P(u) = \sum_{\alpha\in\Gamma^i_d} P_\alpha B^d_\alpha(u)$, where $\{P_\alpha, \alpha \in \Gamma^i_d\}$ is a set of points.*

$\{P_\alpha\}$ is the set of *control* points of the Bézier simplex. This set contains C_i^{i+d} points, for a i-dimensional simplex of degree d: it can be stored in an array of control points, according to the lexicographical order of the multi-indices [22]. The basic idea is the following. Let $\alpha = \alpha_0 \ldots \alpha_i$ be the multi-index of a control point. If $i = 1$ then the point is stored at index α_0, corresponding to the lexicographical order on 1-dimensional multi-indices. If $i > 1$, then it is necessary to store first α_0 $(i-1)$-dimensional sets of control points of degrees decreasing from $\sum_{j=0}^i \alpha_j$ to $1 + \sum_{j=1}^i \alpha_j$, and then to store the point in the same way than if it would belong to the set of control points of a $(i-1)$-dimensional Bézier simplex of degree $\sum_{j=1}^i \alpha_j$: cf. figure $3(c)$.

It is well known that a Bézier simplex β contains its boundary, i.e. the boundary of β is a set of lower dimensional Bézier simplices defined by subsets of control points of β. More precisely, let i be the dimension of β: all control points P_α of β such that the j^{th} component of α is 0 define a $(i-1)$ Bézier simplex. For instance in figure $3(a)$, the points numbered $003, 102, 201, 300$ define a Bézier curve of degree 3.

Consequently, a data structure can be defined, based upon semi-simplicial sets, such that any simplex is associated with its *proper* control points, i.e. control points P_α such that no component of α is null (cf. figure $3(b)$). This set containts C_i^{d-1} points for a i-dimensional simplex of degree d; in fact, the structure of this set corresponds to the structure of a set of control points of a lower dimensional Bézier simplex, and it can be stored in an array as before, by using the lexicographical order of multi-indices.

The set of all control points of a simplex can be retrieved from its proper control points and from the control points of the simplices of its boundary, by taking into account the face operators indices. For instance, the proper points numbered $(21, 12)$ of the edge on the left (resp. right, bottom) of figure 3(b) correspond to the points numbered $(201, 102)$) (resp. $(210, 120)$, $(021, 012)$) in figure 3(a), since the edge on the left correspond to face operator d_1 (resp. d_2, d_0).

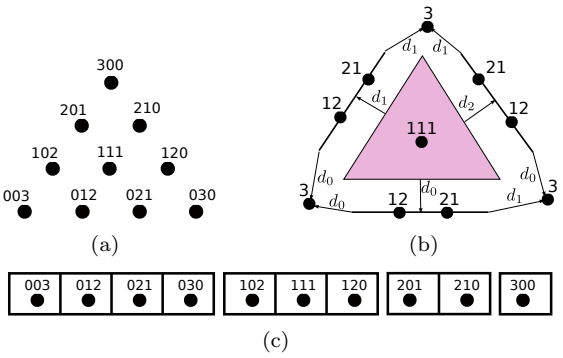

Figure 3: a) The control points of a Bézier triangle of degree 3. b) Each simplex is associated with its proper control points. c) Structure of the control points storage.

Data structure The following data structure is a straightforward implementation of semi-simplicial sets associated with triangular Bézier spaces of degree `degree`. The notation `*Type` denotes an access to a `Type` data ; `[x..y] of Type` denotes an array of `Type` which indices begins at x and ends at y ; `<Type>` denotes a list of `Type` data.

```
type Simplex is record :
   dim :  integer
   faceOps :  [0..dim] of *Simplex
   ctrlPts :  [1..C_dim^(set.degree-1)] of Point
   set :  *SemiSimplicialSet
end record

type SemiSimplicialSet is record :
   dim :  integer
   degree :  integer
   allSimplices :  [0..dim] of <*Simplex>
end record
```

Regarding the `SemiSimplicialSet` structure, it is assumed that all Bézier simplices of a given Bézier space have the same degree, `allSimplices` gives access to the lists of i-simplices, $0 \leq i \leq dim$. Regarding the `Simplex` structure, `faceOps` gives access to the $(dim - 1)$-simplices of its boundary, `set` gives access to its semi-simplicial set and `ctrlPoints` stores its proper control points, ordered by lexicographical order. Note that the multi-index of a point can be retrieved from its index in the array. Obviously, this structure can be adapted. A list of accesses can be added for any i-simplex, in order to access the $(i + 1)$-simplices of its star. The structure can be optimized for different subclasses of semi-simplicial sets, for instance by keeping only n-simplices and by replacing face operators by adjacency relations when n-dimensional triangular manifolds are to be represented [8], etc.

Basic Operations Any semi-simplicial set S can be constructed by applying two basic operations: cone and identification [19].

The cone of S consists in adding a 0-simplex v to S, and in creating a new $(i + 1)$-simplex μ incident to σ and v for any i-simplex σ of S (cf. Figure 4). Proper control points of μ can be computed for instance by linear interpolation between the proper control points of σ and the point associated with v. Note that a complete i-simplex (together with its boundary) is a cone upon a complete $(i - 1)$-simplex (together with its boundary).

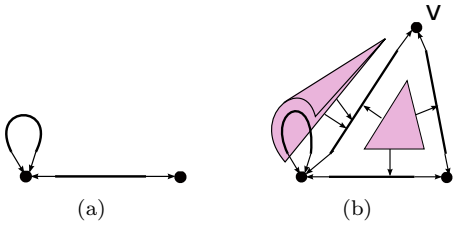

Figure 4: The semi-simplicial set b) can be obtained by a cone operation applied to the semi-simplicial set of a). v is the new vertex.

Two i-simplices σ and τ can be identified if they share the same boundary. This operation consists in replacing σ and τ by a new i-simplex μ such that its boundary is equal to the boundary of σ (and τ), and its star is the union of the stars of σ and τ (more precisely, if $\nu d_j = \sigma$ or $\nu d_j = \tau$ before operation, then $\nu d_j = \mu$ after operation). For instance, the semi-simplicial set of Figure 2(a) can be deduced from the semi-simplicial set of Figure 2(c) by identifying 0-simplices $v1$ and $v2$, 0-simplices $w1$, $w2$ and $w3$, and 1-simplices $a1$ and $a2$. As for the cone operation, the proper control points of new simplices can be computed by linear interpolation of the proper control points of the identified simplices (remember that it is assumed, for a triangular Bézier space, that the degrees of all its Bézier simplices are equal).

Higher-level operations have been defined: for instance, any two i-simplices can be identified, by first identifying their boundaries by increasing dimensions, then by identifying the i-simplices themselves. See also the identification of subsets of simplices, split operation, flip operation, etc. [14].

2.2. Semi-Cubical Sets

Definition 3. *A n-dimensional semi-cubical set $S = (K, (d_j^i))$ is a family of sets $K = (K^p)_{p \in [0..n]}$ together*

with face operators: $d_j^i : K^p \to K^{p-1}$ *for* $1 \leq p \leq n$, $1 \leq i \leq p$ *and* $j \in \{0,1\}$, *which satisfy:* $\sigma d_j^i d_l^k = \sigma d_l^k d_j^{i-1}, \forall i,k$ *such that* $k < i$ *and* $\forall j,l, 0 \leq j,l \leq 1$

An element σ of K^p is a *p-cube*. Intuitively, a p-cube is the cartesian product of p 1-simplices $\sigma_1 \times \ldots \times \sigma_i \times \ldots \times \sigma_p$, and the face operator d_j^i associates σ with the $(p-1)$-cube[3] corresponding to $\sigma_1 \times \ldots \times \sigma_i d_j \times \ldots \times \sigma_p$ (cf. figure 5(a,b)).

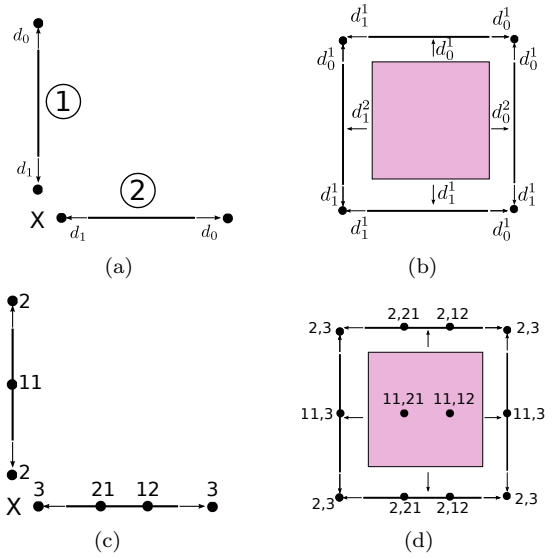

Figure 5: a) and b): intuitive correspondence between the product of two 1-simplices (and their boundaries) and a 2-cube (and its boundary). c) and d) correspondence between control points. For clarity purpose, the face operators are not represented on c), d), since they are the same as in a) and b).

A Bézier patch is the product of Bézier curves, a cubical Bézier space is a set of Bézier patches satisfying some conditions, and the correspondence described above between semi-simplicial sets and triangular Bézier spaces is retrieved here for semi-cubical sets and cubical Bézier spaces: cf. Figure 5(c,d). Note that the degrees of the "curves generating a patch" can be different, and that the control points of a p-dimensional patch are denoted by p-tuples of 1-dimensional multi-indices.

Semi-cubical sets can be easily implemented in a way similar to that described for semi-simplicial sets.

Any semi-cubical set can be constructed by applying two basic operations: extrusion and identification. Extrusion is the particular case of the cartesian product by a 1-simplex (and its boundary); note that any complete p-cube can be constructed by the extrusion of a complete $(p-1)$-cube. Two cubes can be identified if their boundaries are equal, and if their control points have the same structure. Obvioulsy, higher-level operations have been defined: identification and cartesian product of semi-cubical subsets, split, etc.

[3]Note that the cartesian product of a p-cube by a vertex is a p-cube.

2.3. Semi-Simploidal Sets

Combinatorial Structure Semi-simploidal sets generalize semi-simplicial sets and semi-cubical sets, according to the intuitive fact that a simploid is the cartesian product of simplices of any dimensions. A semi-simploidal set is a set of simploids on which act face operators: cf. Figure 6(a,b). As a cube is characterized by its dimension, a simploid σ is characterized by its *type* $(a_1, \ldots, a_n)$. Intuitively, a_i is the dimension of its i^{th} generating simplex, and the face operator d_j^i associates with σ the same product of simplices, except that the i^{th} component is replaced by its j^{th} face.

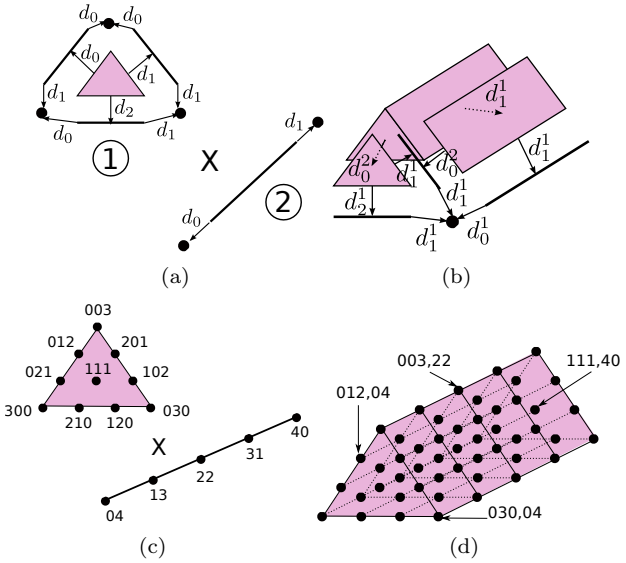

Figure 6: a) and b): intuitive correspondence between the product of two simplices (and their boundaries) and a simploid of type $(2,1)$ (and a part of its boundary). c) and d) correspondence between control points.

The formal definition of semi-simploidal sets is not provided here (cf. [17]): it can be retrieved from the definition of simploidal sets (cf. Section 4), by keeping only face operators. Note that semi-simplicial sets (resp. semi-cubical sets) can be retrieved by assimilating a i-simplex (resp. a i-cube) with a simploid of type (i) (resp. $(1, \ldots, 1)$ of length i, where the length of a type is its number of components).

Bézier Embedding A simploidal Bézier space is a set S of Bézier simploids such that the intersection of two simploids σ and τ is a simploid of S which corresponds (according to the structure and the control point values) to a common face of σ and τ. A Bézier simploid of type $(a_1, \ldots, a_n)$ and degree $(d_1, \ldots, d_n)$ is defined, using control points $\{P_{(\alpha^1,\cdots,\alpha^n)}\}$, by $P(u^1, \ldots, u^n) =$

$$\sum_{\alpha^1 \in \Gamma_{d_1}^{a_1}} \cdots \sum_{\alpha^n \in \Gamma_{d_n}^{a_n}} P_{(\alpha^1,\ldots,\alpha^n)} B_{\alpha^1}^{d_1}(u^1) \times \ldots \times B_{\alpha^n}^{d_n}(u^n)$$

where any u^i is a point of the standard a_i-simplex.

As before, there is a correspondence between the semi-simploidal set structure and the structure of n-tuples of multi-indices: cf. Figure 6(c, d). This correspondence makes it possible to associate its proper control points with any simploid, and to retrieve all control points (and their multi-indices) of a simploid from its proper control points and the proper control points of the simploids of its boundary, using the numbering of the face operators.

As before, control points can be stored into an array, using the lexicographical order of the tuples of multi-indices, which induces a correspondence between the index of a point in the array and its associated tuple of multi-indices [22] (cf. section 4.2). A data structure for implementing semi-simploidal sets embedded as simploidal Bézier spaces can be deduced from the data structure presented in section 4.3.

Basic Operations Basic operations for constructing any semi-simploidal set are the cartesian product and the identification. The cartesian product definition can be deduced from the fact that the cartesian product of two simploids of types $(a_1, \cdots, a_n)$ and $(b_1, \cdots, b_m)$ is a simploid of type $(a_1, \cdots, a_n, b_1, \cdots, b_m)$. The control points of the resulting simploid can be computed for instance by "adding" the control points (i.e. their coordinates) of the initial simploids (Minkowski sum). For instance on Figure 6(d), the point corresponding to $(030, 04)$ can be defined as the "sum" of the points (030) and (04) of Figure 6(c). As before, two simploids can be identified if they share the same boundary, and if their control points have the same structure.

3. Simplicial

This section is mainly based upon [2, 10]. Our main contribution here is the relation between degeneracy operators and control points.

Combinatorial Structure The cartesian product operation is not directly defined for semi-simplicial sets, but it is defined in a very simple way on simplicial sets, thanks to the notion of *degenerate* simplex. This notion makes it also possible to get simplex shapes which are slightly more general than usual: cf. Figure 7(b).

Definition 4. *(cf. Figure 7) A* ***simplicial set*** $S = (K, (d_j), (s_j))$ *is a family* $K = \left(K^i\right)_{i\in\mathbb{N}}$ *of simplices together with two families of maps:*
• face operators*:* $\forall i > 0$, $\forall j$, $0 \leq j \leq i$, $d_j : K^i \to K^{i-1}$,
• degeneracy operators*:* $\forall i \geq 0$, $\forall j$, $0 \leq j \leq i$, $s_j : K^i \to K^{i+1}$,
satisfying: $d_j d_l = d_l d_{j-1}$ *if* $l < j$, $s_j s_l = s_l s_{j+1}$ *if* $l \leq j$, $s_j d_l = d_l s_{j-1}$ *if* $l < j$, $s_j d_l = d_{l-1} s_j$ *if* $l > j+1$ *and* $s_j d_j = s_j d_{j+1} = Id$.

An $(i+1)$-simplex μ is *degenerate* if $\mu = \sigma s_j$ for some σ and some j. Intuitively, if a i-simplex σ is assimilated with a sequence of $i+1$ vertices $(v_0, \cdots, v_j, \cdots, v_i)$, σs^j corresponds to $(v_0, \cdots, v_j, v_j, \cdots, v_i)$.

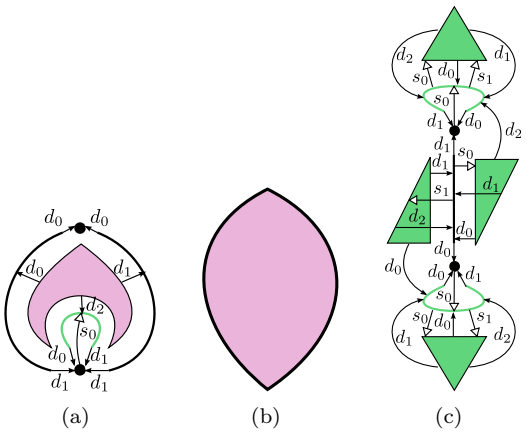

Figure 7: *a*) a simplicial set; only the degenerate simplex which is a face of a non degenerate simplex is represented; and *b*) a geometric representation *c*) an edge (and its boundary) and all degenerate simplices up to dimension 2.

From a theoretical point of view, all denererate simplices deduced from a simplex by a sequence of degeneracy operators exist in a simplicial set: thus any simplicial set contains an infinite number of degenerate simplices. For instance, the existence of all these degenerate simplices makes it possible to get a very simple definition of the cartesian product (cf. below). But from a practical point of view, only the explicit representation of non degenerate simplices and their faces (degenerate or not) is required (cf. figure 7(a, b)); the other simplices, which are degenerate, can be implicitly handled. Indeed, any degenerate simplex can be obtained by applying a sequence of degeneracy operators to a non degenerate simplex. So, the degenerate simplices which appear only in the boundary of degenerate simplices can be implicitly handled, through the corresponding non degenerate simplex and the sequence of indices of the corresponding degeneracy operators. Compared with the implementation of semi-simplicial sets, the additional cost of the implementation of simplicial set is thus low, and related to the object which has to be represented. Obviously, this implicit representation of most degenerate simplices has to be taken into account when implementing the construction operations.

Bézier embedding Let σ be any i-simplex of a simplicial set. From a theoretical point of view, and as for semi-simplicial sets, a Bézier simplex is associated with σ. The relation between the control points associated with σ and with σd_j, for any j, is the relation described for semi-simplicial sets. We define the following new relation for degeneracy operators:

Constraint 1. *Let* σ *be a* i*-simplex and* $\mu = \sigma s_j$*, for any*

j. Let $P = P_{\alpha_0 \cdots \alpha_i}$ *be a control point of* σ*; then all points* $P_{\alpha'_0 \cdots \alpha'_{i+1}}$ *of* μ *such that :*

- $\alpha'_l = \alpha_l$, $0 \leq l \leq j-1$
- $\alpha'_j + \alpha'_{j+1} = \alpha_j$, $0 \leq \alpha'_j, \alpha'_{j+1} \leq \alpha_j$
- *and* $\alpha'_l = \alpha_{l-1}$, $j+2 \leq l \leq i+1$

are equal to P*.*

Constraint 2. *Let* $P = P_{\alpha_0 \cdots \alpha_i}$ *be a control point of* σ*; then all points* $P_{\alpha'_0 \cdots \alpha'_{i+1}}$ *of* μ *such that :*

- $\alpha'_l = \alpha_l$, $1 \leq l \leq j-1$
- $\alpha'_j + \alpha'_{j+1} = \alpha_j$, $0 \leq \alpha'_j, \alpha'_{j+1} \leq \alpha_j$
- *and* $\alpha'_l = \alpha_{l-1}$, $j+2 \leq l \leq i+1$

are equal to P*.*

For instance on Figure 8(a), all proper control points of the degenerate edge αs_0 (resp. γs_0) are equal to control point P_4 of vertex α (resp. γ). Proper control points P_{211} and P_{121} (resp. P_{112}) of the degenerate edge βs_0 are equal to control point P_{31} (resp. P_{22}) of β. Similarly, proper control points P_{121} and P_{112} (resp. P_{211}) of degenerate simplex βs_1 are equal to control point P_{13} (resp. P_{22}) of β. So, the shape of a degenerate simplex σs_j is the shape of σ, and a triangular Bézier space can be associated with any simplicial set.

From a practical point of view, only non degenerate simplices are associated with their proper control points. The control points of any simplex (degenerate or not) can be retrieved from these control points and from the relations corresponding to face and degeneracy operators: in particular, when the simplex is degenerate, *constraint 1* makes it possible to retrieve its control points among the control points associated with its corresponding non degenerate simplex. So, the additional cost for embedding simplicial sets, compared with semi-simplicial sets, is null, except the computing time for accessing the control points of a degenerate simplex: in this case, the access time is lower than the dimension of the simplex, with an adequate data structure (cf. section 4.3). Note that, as for the semi-simplicial case, it is assumed that the degrees of all Bézier simplices are equal, whenever they are degenerate or not.

Basic Operations Basic operations are still cone and identification. The cone definition is a direct extension of the cone of a semi-simplicial set. The identification of two i-simplices σ and μ, such that their boundaries are equal, implies now the identification two by two of all degenerate simplices deduced from σ and μ by a sequence of degeneracy operators. Note that identifying a non degenerate simplex with a degenerate simplex produces a degenerate simplex [10].

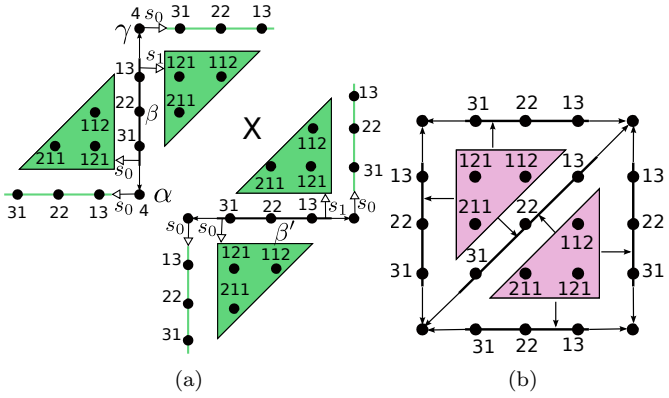

Figure 8: Correspondence between control points.

The cartesian product of two simplicial sets S and S' can be easily defined: a i-simplex μ from $S \times S'$ is associated with any pair (σ, σ') of i-simplices of S and S', such that for any j, $\mu d_j = (\sigma d_j, \sigma' d_j)$ and $\mu s_j = (\sigma s_j, \sigma' s_j)$ (cf. Figure 9). The control points associated with μ can be computed by "adding" the corresponding points of σ and σ' (Minkowski sum). For instance on Figure 8(b), point P_{121} of $(\beta s_1, \beta' s_0)$ corresponds to the sum of P_{13} of β (and thus P_{121} of βs_1) and P_{31} of β' (and thus P_{121} of βs_0). In practice, only the degenerate simplices having a dimension lower than the sum of the dimensions of S and S' are needed for computing the cartesian product of S and S'. At last, note that it is possible to directly deduce the cartesian product of two semi-simplicial sets by implicitly handling the degenerate simplices (and thus the degeneracy operators): cf. [23].

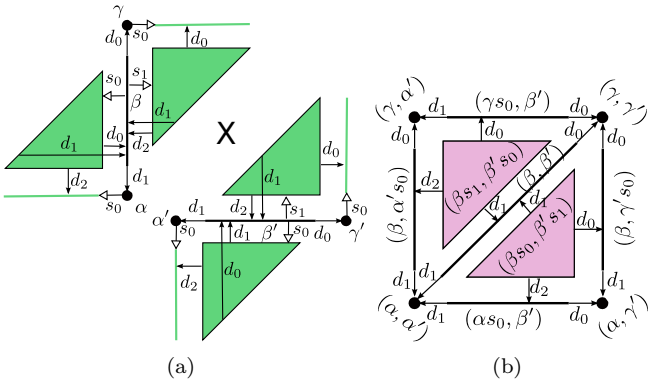

Figure 9: b) the non degenerate simplices of the cartesian product of the two edges of a), represented with the degenerate simplices useful for their cartesian product.

4. Simploidal

As seen before, semi-simploidal sets contain and generalize semi-simplicial sets and semi-cubical sets, but not the

simplicial sets and cubical sets. For instance, the cone operation defined for semi-simplicial sets cannot be defined for semi-simploidal sets.

Similarly to simplicial and cubical sets[4], simploidal sets are defined by adding *degeneracy operators* to semi-simploidal sets: cf. Figure 10(a, b). In particular, this makes it possible to define the cone operation as an extrusion followed by a degeneracy: cf. Figure 10(c).

4.1. Combinatorial Structure

Definition 5. *A simploidal set* $S = (K, (d^i_j), (s^i_j))$ *is a set of* simploids *equipped with a* type *operator* $\mathcal{T} : K \mapsto \bigcup_{i=0}^{\infty} \mathbb{N}^{*i}$, *face operators* d^i_j *and degeneracy operators* s^k_l.

Let $\sigma \in K$; $\sigma\mathcal{T}$ *is the* type *of* σ. *Let* $\sigma\mathcal{T} = (a_1, \cdots, a_n)$: σd^i_j *(resp.* σs^k_l*) is defined if* $1 \leq i \leq n, 0 \leq j \leq a_i$ *(resp.* $0 \leq k \leq n$ *and* $l = -1$, *or* $1 \leq k \leq n$ *and* $0 \leq l \leq a_k$*). Operators satisfy:*

(I) Action on the type

$$1)\ \sigma d^i_j\mathcal{T} = \begin{cases} (a_1, ..., a_i - 1, ..., a_n)\ \textit{if } a_i > 1 \\ (a_1, ..., a_{i-1}, a_{i+1}, ..., a_n)\ \textit{otherwise} \end{cases}$$

$$2)\ \sigma s^i_j\mathcal{T} = \begin{cases} (a_1, ..., a_i + 1, ..., a_n)\ 1 \leq i \leq n, 0 \leq j \leq a_i \\ (a_1, ..., a_i, 1, a_{i+1}, ..., a_n)\ \ 0 \leq i \leq n, j = -1 \end{cases}$$

(II) Commutation of face operators

$$1)\ d^i_j d^i_l = d^i_l d^i_{j-1} \quad l < j, a_i > 1$$

$$2)\ d^i_j d^k_l = \begin{cases} d^k_l d^i_j & \textit{if } a_k > 1 \\ d^k_l d^{i-1}_j & \textit{otherwise} \end{cases} \qquad k < i$$

(III) Commutation of degeneracy operators

$$1)\ s^i_{-1} s^k_{-1} = s^k_{-1} s^{i+1}_{-1} \quad k \leq i$$

$$2)\ \textit{if } l \neq -1, s^i_{-1} s^k_l = \begin{cases} s^k_l s^i_{-1} \textit{if } k \leq i \\ s^{k-1}_l s^i_{-1} \textit{if } i < k - 1 \end{cases}$$

$$3)\ \textit{if } j, l \neq -1, s^i_j s^k_l = \begin{cases} s^k_l s^i_j \textit{if } i \neq k \\ s^k_l s^i_{j+1} \textit{if } i = k, l \leq j \end{cases}$$

(IV) Commutation of face and degeneracy operators

$$1)\ \textit{if } l \neq -1, s^i_l d^i_j = \begin{cases} d^i_j s^i_{l-1} & \textit{if } j < l \\ d^i_{j-1} s^i_l & \textit{if } j > l + 1 \\ s^i_l d^i_{j+1} = Id & \textit{if } j = l \end{cases}$$

$$2) \textit{if } l = -1, s^{i-1}_l d^i_j = Id$$

In the other cases:

$$3)\ \textit{if } k \leq i - 1, s^k_l d^i_j = \begin{cases} d^i_j s^k_l & \textit{if } l \neq -1 \\ d^{i-1}_j s^k_l & \textit{otherwise} \end{cases}$$

$$4)\ \textit{else (i.e. } k \geq i), s^k_l d^i_j = \begin{cases} d^i_j s^k_l & \textit{if } a_i > 1 \\ d^i_j s^{k-1}_l & \textit{otherwise} \end{cases}$$

Let σ be a simploid of type $(a_1, ..., a_n)$: n is the *length* of σ, and its *dimension* is $\sum_{i=1}^{n} a_i$, or 0 if $n = 0$. Intuitively, a simploid is either a vertex, or the cartesian product of n "generating simplices" of respective dimensions $a_1, ..., a_n$, with $a_i > 0$ for any i. No "0" appears in the type of a simploid, since the cartesian product by a vertex is equal to the identity, from a structural point of view. This explains the fact that several cases are distinguished in the definition of simploidal sets, even if these cases are intuitively similar. More generally, this definition can be retrieved from the previous remark, the properties of face and degeneracy operators of simplicial sets, and from the fact that the actions of two operators on two different "generating simplices" are independent.

Equations (I) denotes the action of an operator on the simploid type (equations (I-1) for a face operator, equations (I-2) for a degeneracy operator). The action of a face (resp. degeneracy) operator on a "generating simplex" decreases (resp. increases) its dimension; hence, if a zero appears after the application of a face operator (i.e. if $a_i = 1$ and d^i_j is applied), it is removed from the type. Conversely, the application of degeneracy operator s^k_{-1} intuitively consists in adding a degenerate edge as new "generating simplex".

Equations (II) (resp. (III), (IV)) corresponds to the commutation relation of face operators (resp. degeneracy operators, degeneracy and face operators). Several cases correspond to the commutation properties of simplicial sets, i.e. when operators are applied on the same "generating simplex": cf. equation (II-1), the last case of equation (III-3), equation (IV-1). Equation (IV-2) corresponds to the removal, by a face operator, of a new 1-dimensional "generating simplex" issued from a degeneracy operator (i.e. nothing is modified).

The other cases correspond to the independence of the actions of operators applied to distinct "generating simplices":

- equations (II-2) : two cases are distinguished, according to the fact that a 1-dimensional "generating simplex" disappears when a face operator is applied;
- equations (III-1), (III-2) and the first case of equation (III-3) : these cases are distinguished, according

[4] Cubical sets are not recalled here: see [11, 12]

to the number of new "generating simplices" created by operator s_{-1}; note that there is no commutation property in equation (III-2) when $k = i + 1$, since $\sigma s_{-1}^{k-1} s_l^k \mathcal{T} = (a_1, \ldots, a_{k-1}, 2, \ldots, a_n)$ (the first degeneracy operator creates a new 1-dimensional "generating simplex", on which another degeneracy operator is applied);

- equations (IV-3) and (IV-4): these cases are distinguished, according to the creation or removal of 1-dimensional "generating simplices".

The definitions of simplicial sets (resp. cubical sets) can be retrieved when the types of i-dimensional simploids contain only one component (resp. only components equal to 1), for $i > 0$.

Even if the definition of simploidal sets above contains many properties, it easy to implement, contrary to the simpler definition of supercomplexes proposed by Gugenheim in 1957, for topological purposes (cf. section 4.5 for a comparison of simploidal sets and supercomplexes).

Simploidal sets are also efficient, since the properties of the definition have not to be implemented explicitly. These properties have to be satisfied by the construction operations, and this involves no particular cost. More precisely, and as for simplicial sets, only non degenerate simploids and theirs faces (degenerate or not) have to be explicitly represented. So, the additional cost of simploidal sets is low, even compared with semi-simplicial sets; it is related to the representation of the type of any simploid, and to the "complexity" of the object which has to be represented (cf. section 4.3).

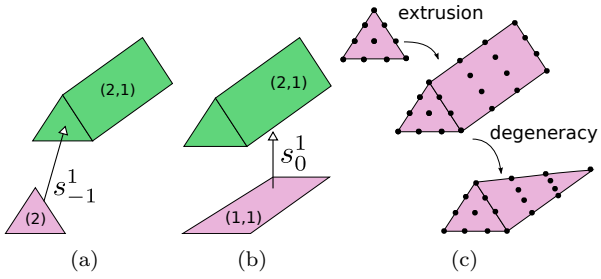

Figure 10: A simploid of type $(2, 1)$ can be degenerated into a simploid of type (2) (cf. *a*)), or $(1, 1)$ (cf. *b*)). The cone operation consists in an extrusion followed by a degeneracy. On *c*), the control points of a square face are displayed before and after the degeneracy: all control points of the degenerate edge are equal, but the control point structure of the resulting face is not that of a triangle.

4.2. Bézier embedding

Associating a simploidal set with a simploidal Bézier space can be done by extending the results described for the simplicial case. More precisely, if $P_\sigma = \{P_{(a^1,\ldots,a^n)}, a^i \in \Gamma_{d_i}^{a_i}, 1 \leq i \leq n\}$ is the set of control points of a simploid σ of type $(a_1, ..., a_n)$, then:

- Any control point $P_{(a^1,\ldots,a^i,\ldots,a^n)}$, such that the j^{th} component of a^i is 0, is equal to the control point $P_{(a^1,\ldots,a'^i,\ldots,a^n)}$ of σd_j^i, where a'^i equals a^i without its $j^{th}eme$ component ;

- All control points $P_{(a^1,\ldots,a'^i=a_0'^i\ldots a_j'^i a_{j+1}'^i\ldots a_{a_i+1}'^i,\ldots,a^n)}$ of σs_j^i are equal to the control point $P_{(a^1,\ldots,a^i=a_0^i\ldots a_j^i\ldots a_{a_i}^i,\ldots,a^n)}$ of σ, such that $a_k'^i = a_k^i$ for $k < j$, $a_j'^i + a_{j+1}'^i = a_j^i$, $a_k'^i = a_{k-1}^i$ for $k > j + 1$.

In practice, it is sufficient to associate its proper control points with any non degenerate simploid. The set of control points of any simploid (and so the associated Bézier simploid) can be reconstructed according to the previous relations:

- if the simploid is a vertex, a point is associated with it;

- else, if the simploid is not degenerate, its associated control points can be retrieved from its proper control points, and from the proper control points of the simploids of its boundary, using the numbering of face operators;

- else, if the simploid is degenerate, it is necessary to access the control points of its associated non degenerate simploid, using the numbering of degeneracy operators and the relation described above, which links the control points of simploids σ and σs_j^i.

.

Remember also that the computation of all Bézier simploids of a simploidal set can be useless, depending on the application: for example, only main simploids are necessary in order to display a simploidal set.

At last, note that control points of non degenerate simploids can be stored into an array according to the lexicographical order of tuples of multi-indices [22]. For instance, let $P = P_{(a^1,\ldots,a^n)}$ be a control point of a Bézier simploid of type $(a_1, \ldots, a_n)$ and degree $(d_1, \ldots, d_n)$. If $n = 1$, then the simploid corresponds to a simplex, and P is stored in the way described in section 2.1. If $n > 1$, it is necessary to store all control points of a simploid of type $(a_2, \ldots, a_n)$ of degree $(d_2, \ldots, d_n)$ for each multi-index preceding $a^1 = (\alpha_0, \ldots, \alpha_{a_1})$; then point P can be stored as if it would be a control point of a Bézier simploid of type $(a_2, \ldots, a_n)$ and degree $(d_2, \ldots, d_n)$, with a tuple of multi-indices equal to $(a^2, \ldots, a^n)$.

4.3. Data Structure

The following data structure is a possible implementation of simploidal sets associated with simploidal Bézier spaces.

```
type Simploid is record :
  n :  integer
  type :  [1..n] of integer
  degen :  boolean
```

```
   faceOps :  [0..(∑_{i=1}^{n}(a_i + 1)) − 1] of *Simploid
   degenOps :  *[0..n + ∑_{i=1}^{n}(a_i + 1)] of *Simploid
   ctrlPts :  *[1..C_{a_1}^{d_1−1} × ... × C_{a_n}^{d_n−1}] of CtrPt
   set :  *SimploidalSet
end record

type SimploidalSet is record :
   dim :  integer
   degree :  integer
   allSimploids :  [0..dim] of <*Simploid>
end record
```

Regarding the `SimploidalSet` structure, `dim` is the dimension of the simploidal set, `allSimploids` stores the lists of simploids, ordered by increasing dimensions; `degree` is the degree of each "generating simplex". It is thus assumed here that, whatever its type, the degree of a simploid of length n is ($d_1 =$ `degree`$, \ldots, d_n =$ `degree`). In this way, the identification of two simploids is simplified, as they have for sure the same control point structure.

Regarding the `Simploid` structure, let σ be a simploid of type $(a_1, ..., a_n)$ and degree $(d_1, \ldots, d_n)$. `set` gives access to its simploidal set. `n` is the length of its `type`, a_i corresponds to σ`.type[i]` and d_i corresponds to σ`.set.degree`.

Let $d = \sum_{i=1}^{\sigma.n} \sigma.type[i]$ be the dimension of σ. `faceOps` gives access to the boundary simploids of dimension $d-1$.

`degen` is true if σ is degenerate: in this case, `ctrlPts` is *null*; in the other case, `ctrlPts` gives an access to the array containing the proper control points of σ, ordered by lexicographical order. The n-tuple of multi-indices of a point can be retrieved from its index in the array.

Remember that only simploids which are in the boundaries of non degenerate simploids are explicitly represented. So, the degeneracy operators can be not explicitly defined for some simploids. If it is the case for σ, then `degenOps` is *null*; else, `degenOps` is an access to an array such that each entry corresponds to a possible degeneracy operator; and if this operator is not explicitly represented, the corresponding entry is *null*. Another solution would be to associate a list of accesses with each simploid, together with the corresponding numberings, in order to represent only explicit degeneracy operators. The respective interests of one solution compared with the other depend on the objects which are represented.

Note that it could be interesting to access all simploids of dimension $d+1$ in the star of σ; these accesses corresponds to the inverse of the face operators, and can be represented by a list of $*Simploid$ associated with each simploid.

If σ is degenerate, it is important to access the corresponding non degenerate simploid τ. In fact, τ is in the boundary of σ, and can be accessed by using the face operators, since there are relations between face and degeneracy operators (for the same reason, the information $\sigma.degen$ is redundant, since it can be retrieved by checking if a degeneracy operator of a face of σ gives access to σ). In order to optimize the access to τ, i.e. to avoid to check all face operators, it can be interesting to associate with σ a list of numberings, corresponding to the face operators accessing the face from which σ is degenerated.

The structure can be optimized for different subclasses of simploidal sets, for instance for semi-simploidal sets, cubical sets (only the dimensions of cubes are required, since the type of a d-dimensional cube is the d-tuple $(1, \ldots, 1)$). Other optimizations can be proposed, for instance for manifolds: all simploids have the same dimension, and face operators are replaced by adjacency operators.

At last, note that the additional cost, compared with a similar structure implementing simplicial sets, is low, since it corresponds to the types which have to be explicitly associated with all simploids: it is thus at most the dimension of the simploidal set times the number of simploids.

4.4. Basic Operations

From a theoretical point of view, and similarly to the simplicial case, all degenerate simploids associated with any simploid exist in a simploidal set; so, a simploidal set contains an infinite number of simploids. In practice, only the non degenerate simploids and their faces (degenerate or not) are needed. For some operations, other degenerate simploids are useful, but they can be implicitly handled. For instance, identifying a simploid with a degenerate simploid results in degenerating the first simploid: this can be done directly by modifying a degeneracy operator.

Cartesian Product As for semi-simploidal sets, the definition of the cartesian product of simploidal sets is deduced from the fact that the type of the cartesian product ν of two simploids σ and μ of type $(a_1, ..., a_n)$ and $(b_1, ..., b_m)$ is $(a_1, ..., a_n, b_1, ..., b_m)$. For example, νd_j^i corresponds to the product of σd_j^i (resp. σ) and μ (resp. μd_j^{i-n}) if $i \leq n$ (resp. $i > n$); similarly, νs_j^i corresponds to the product of σs_j^i (resp. σ) and μ (resp. μs_j^{i-n}) if $i \leq n$ (resp. $i > n$), with the particularity: the product of σs_{-1}^n and μ is identified with the product of σ and μs_{-1}^0, and corresponds to νs_{-1}^n.

This operation can be easily extended in order to define the cartesian product of subsets of given simploidal sets. Note also that the "simplicial " cartesian product can be applied when the subsets contain only simplices (i.e. simploids equivalent to simplices).

Identification As for semi-simploidal sets, two simploids σ and μ can be identified under some structural conditions (equality of type and boundary) and geometrical conditions (equality of degree). Regarding face operators, identification acts in the same way as for semi-simploidal case. Regarding degeneracy operators, the identification of σ and μ induces the identifications of their degenerated simploids (this is similar to the simplicial case). The computation of the control points of the simploid resulting from the identification of σ

and μ can be done as in the semi-simploidal framework. This operation can be generalized in order to identify simploidal subsets: the identification of simploids induces then the identification of their boundaries and their degenerate simploids.

Degeneracy Degenerating a simploid σ into a simploid μ of its boundary simply consists in identifying σ with a degenerate simploid associated with μ by a sequence of degeneracy operators. Of course, the boundary of σ has to satisfy some conditions, which can be deduced from the definition of simploidal sets. This operation can be generalized in order to degenerate a simploidal subset S into a single vertex: this can be achieved by identifying all vertices of S, and then by iteratively identifying all edges and in degenerating the resulting edge, then in applying the same process to all 2-dimensional simploids (according to their types), and so on.

Note that simploids exist, which have similar "shapes" but different types. Two different cases can be distinguished. The first one comes from the fact that a sequence can be different from one of its permutations. For instance, a simploid of type $(1, 2)$ is not a simploid of type $(2, 1)$ (the cartesian product is not commutative). A similar remark can be stated for simplicial sets, since simplicial sets exist, which are not isomorphic, but their geometric realizations are isomorphic. The second one comes from particular cases of degeneracy: for instance, a quadrangle, i.e. a simploid of type $(1, 1)$, looks like a triangle, i.e. a simploid of type (2), when one of its edge is degenerate: but the difference is clear when one looks at the structures of control points, which are not isomorphic: cf. Figure 10(c).

Cone The extrusion of a simploidal set S is the cartesian product of S with another simploidal set containing three non degenerate simploids: an edge and the two vertices of its boundary. The resulting simploidal set therefore contains two"copies" of S, corresponding to the product of S with the two vertices, and a set of simploids which "links" the two copies of S. From a theoretical point of view, defining the cone of a simploidal set S consists in extruding S, and then in degenerating a resulting copy of S into a vertex (cf. the degeneracy operation above and Figure 10(c)). Of course, this definition can be optimized. As for the cartesian product, it is possible to define the cone operation for a simploidal subset.

Obvously, other operations can be defined, based on these basic operations, in order to construct simploidal sets.

4.5. Discussion about Gugenheim's definition

Supercomplexes were defined in 1957 (cf. [18] page 37). By mimicking the definition of simploidal sets, (the structure of) supercomplexes could be defined in the following way:

Definition 6. *A supercomplex $S = (K, (d^i_j), (s^i_j))$ is a set of simploids equipped with a type operator $\mathcal{T} : K \mapsto \bigcup_{i=1}^{\infty} \mathbb{N}^i$, face operators d^i_j and degeneracy operators s^k_l. Let $\sigma\mathcal{T} = (a_1, \cdots, a_n)$: σd^i_j (resp. σs^k_l) is defined if $1 \leq i \leq n, a_i > 0, 0 \leq j \leq a_i$ (resp. $1 \leq k \leq n$ and $0 \leq l \leq a_k$). Operators satisfy:*

- $\sigma d^i_j \mathcal{T} = (a_1, ..., a_i - 1, ..., a_n)$,
  $\sigma s^i_j \mathcal{T} = (a_1, ..., a_i + 1, a_{i+1}, ..., a_n)$;
- *face and degeneracy operators having the same exponent satisfy the commutation properties of face and degeneracy operators of simplicial sets;*
- *face and degeneracy operators having different exponents commute.*

Obviously, this definition of supercomplexes is simpler than the definition of simploidal sets given in 4.1. This is due to the fact that the type of a simploid may contain components equal to 0. Let σ be a supercomplex of type $(2, 1)$ (i.e a prism), then the type of σd^2_0 is $(2, 0)$ (i.e. a triangle). Conversely, in order to associate a degenerate cube with a square, the type of the square has to be for instance $(1, 1, 0)$, so the degeneracy operator s^2_0 can be applied. In fact, the simple definition of supercomplexes is based on the fact that, implicitly, an equivalence relation exists between types which differ only by components equal to 0: this corresponds to the fact that the cartesian product by a vertex is (structurally speaking) equal to the identity. Note that Gugenheim was not interested in the definition of embedding, construction operations nor implementation, and that this equivalence relation was never explicitly stated. This can have some consequences for the conception of data structures and/or operations. For instance, assume a data structure is conceived for implementing supercomplexes. It is necessary to explicitly represent its type for each simploid, but the types of two equivalent simploids can differ by components equal to 0. So, when one intend to identify two equivalent simploids having the same boundary, it is necessary to check the correspondence of the types; this is slightly more complicated than checking their equality. It is easy to find more elaborated examples, showing that the implementation of supercomplexes is more complex than that of simploidal sets (i.e. additional computations have to be performed), due to this implicit equivalence relation between types of simploids.

In order to warrant the unicity of the types, our definition of simploidal sets contains more constraints over boundary and degeneracy operators. Since no 0 appears in the type of a simploid, many cases have to be distinguished in the definition; but, as said above, this does not involve any additional cost, since the properties of the operators don't involve any requirement in space nor in computation. The properties described by equations (I) to (IV) are taken into account by the modifications which are performed during the application of an operation, but

they don't involve a particular computation. Even if it would be necessary, checking that the properties of a simploidal set are satisfied would not involve additional space or time requirements, compared with supercomplexes.

So, our definition seems more efficient regarding implementation issues, and the definition of relations with simploidal Bézier spaces as that of construction operations is quite simple, compared with similar operations defined on simplicial and cubical sets. The counterpart is the fact that the commutation properties of the face and degeneracy operators cannot be expressed as simply as in the supercomplexes definition.

5. Conclusion

Simploidal sets generalize simplicial sets and cubical sets: they are defined in a similar way, by abstract simploids on which act face and degeneracy operators. All basic operations defined in the simplicial and cubical frameworks can be defined for simploidal sets. Relations between simploidal sets and simploidal Bézier spaces generalize similar relations between simplicial (resp. cubical) sets and triangular (resp. cubical) Bézier spaces. The framework presented in this paper is thus general and coherent. This means that it is possible to conceive softwares implementing simploidal sets (e.g. for computing topological properties or for geometric modeling) which make it possible to handle also simplicial sets or cubical sets as particular cases; moreover, it is possible to handle simultaneously simplices and cubes (for instance, a cube and a tetrahedron can be glued with a prism, which is a simploid). The cost of this generalization is low, since the definition of simploidal sets is very close to the definitions of simplicial and cubical sets: the main difference, when implementing simploidal sets, corresponds to the fact that the type of a simploid has to be explicitly represented.

Note that a difference exists between semi-simplicial sets and semi-simploidal sets. It is possible to directly adapt the definition of the cartesian product of simplicial sets for semi-simplicial sets, by *implicitly* handling degeneracy operators [23]: this comes from the fact that the cartesian product of any two simplices (and their boundaries) can be described as a semi-simplicial set. But it is not possible to directly adapt the definition of the cone of simploidal sets for semi-simploidal sets, because the cone of a simploid is not necessarily a simploid.

The relations between simploidal sets and simploidal Bézier spaces have been stated. Note that simploidal sets can be linearly embedded, as a particular case; only vertices are associated with points. We intend to study generalizations, associating more general splines with simploids.

It has been shown that degenerate simploids can have similar shapes, for instance a square in which an edge is degenerated into a vertex, and a triangle; but their structures are different, so it is not possible to glue four tetrahedra on the four "triangle-shaped" faces of a pyramid represented by a prism in which an edge is degenerate, since only two faces are "true" triangles. Similarly, in the proposed SimploidalSet data structure, a single common degree is set for each possible generating simplex, but this structure can be generalized by allowing each generating simplex to have its own degree. In this case, even if two simploids have same type, the structure of their control points may be different. To tackle these two problems, we are studying equivalence relations between simploids having "similar shapes", in order to make these gluings possible.

At last, we intend to study other construction operations, since fundamental differences exist between simplicial structures and simploidal structures: for instance, when an edge is split in a simplicial set, all simplices incident to this edge have to be split, but this operation remains a local one, since only the star of the edge is modified. When an edge is split in a simploidal set, it is necessary to split the simploids of the star of the edge, but this can induce the split of other edges, and thus of other simploids: the operation is not a local one in this case. It will be also interesting to study the computation of topological properties: for instance, since simploids correspond to cartesian products of simplices, classical methods for computing the homological information can be adapted for semi-simploidal sets, and thus for simploidal sets [17].

Acknowledgements The authors gratefully acknowledge Sylvie Alayrangues and Laurent Fuchs for their encouragement and help.

[1] M. K. Agoston, Algebraic Topology, a first course, Pure and applied mathematics, Marcel Dekker Ed., 1976.

[2] J. P. May, Simplicial Objects in Algebraic Topology, Van Nostrand, 1967.

[3] J. R. Munkres, Elements of algebraic topology, Perseus Books, 1984.

[4] W. S. Massey, A basic course in algebraic topology, Springer-Verlag, 1991.

[5] H. Edelsbrunner, H. Harer, Computational Topology - an Introduction, American Mathematical Society, 2010.

[6] J. Rubio, F. Sergeraert, Constructive homological algebra and applications, http://arxiv.org/abs/1208.3816 (August, 28 - September, 02 2006).

[7] J.-G. Dumas, F. Heckenbach, B. D. Saunders, V. Welker, Computing simplicial homology based on efficient smith normal form algorithms., in: Algebra, Geometry, and Software Systems, 2003, pp. 177–206.

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

[9] P. Frey, P.-L. George, Mesh Generation: application to finite elements, Wiley, 2008.

[10] V. Lang, P. Lienhardt, Simplicial sets and triangular patches, in: Computer Graphics International Conference, CGI 1996, Pohang, Korea, June 24-28, 1996, 1996, pp. 154–163.

[11] J.-P. Serre, Homologie singulière des espaces fibrés, The Annals of Mathematics 54 (3) (1951) 425–505.

[12] R. Brown, P. J. Higgins, On the algebra of cubes, Journal of Pure and Applied Algebra 21 (3) (1981) 233 – 260.

[13] T. Kaczynski, K. Mischaikow, M. Mrozek, Computational Homology, Applied Mathematical Sciences, Springer, 2004.

[14] B. Bechmann, D. et Péroche, Informatique graphique, modélisation géométrique et animation, Signal et Image, Traité IC2, Lavoisier, 2007.

[15] F. Ledoux, J. Shepherd, Topological and geometrical properties of hexahedral meshes, Eng. Comput. (Lond.) 26 (4) (2010) 419–432.
[16] W. Dahmen, C. A. Micchelli, On the linear independence of multivariate b-splines I. Triangulation of simploids, SIAM J. Numer. Anal. 19.
[17] S. Peltier, L. Fuchs, P. Lienhardt, Simploidals sets: Definitions, operations and comparison with simplicial sets, Discrete App. Math. 157 (2009) 542–557.
[18] V. K. A. M. Gugenheim, On supercomplexes, Transactions of the American Mathematical Society 85 (1) (1957) 35–51.
[19] V. Lang, P. Lienhardt, Geometric modeling with simplicial sets, in: T. K. S.Y. Shin (Ed.), Computer Graphics and Applications, Pacific Graphics, World Scientific Publishing, Seoul, Corea, 1995, pp. 475–494.
[20] S. Eilenberg, J. Zilber, Semi-simplicial complexes and singular homology, Annals of Mathematics 51 (1950) 499–513.
[21] A. Hatcher, Algebraic Topology, Cambridge University Press, 2002.
[22] T. DeRose, R. N. Goldman, H. Hagen, S. Mann, Functional composition algorithms via blossoming, Transactions On Graphics 12 (2) (1993) 113–135.
[23] P. Lienhardt, X. Skapin, A. Bergey, Cartesian product of simplicial and cellular structures, Int. J. Comput. Geometry Appl. 14 (3) (2004) 115–159.