



HAL
open science

Tubular parametric volume objects: Thickening a piecewise smooth 3D stick figure

Samuel Peltier, Géraldine Morin, Damien Aholou

► To cite this version:

Samuel Peltier, Géraldine Morin, Damien Aholou. Tubular parametric volume objects: Thickening a piecewise smooth 3D stick figure. *Computer Aided Geometric Design*, 2021, 85, pp.101981. 10.1016/j.cagd.2021.101981 . hal-03180193

HAL Id: hal-03180193

<https://hal.science/hal-03180193>

Submitted on 24 Apr 2023

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.



Distributed under a Creative Commons Attribution - NonCommercial 4.0 International License

Tubular Parametric Volume Objects: Thickening a Piecewise Smooth 3D Stick Figure

Samuel Peltier^a, Géraldine Morin^b, Damien Aholou^a

^aUniversité de Poitiers, Laboratoire XLIM, Axe ASALI, CNRS 7252, France

^bUniversité de Toulouse, Laboratoire IRIT, France

Abstract

In this paper, a volume parametric model is computed from a piecewise smooth skeleton. Generating a volume model from a stick figure S defined in 3D is an intuitive process: given S whose topology is a pseudo-graph and whose edges are embedded as Bézier curves in \mathbb{R}^3 , we propose a method for creating a thick volume parametric model "around" S . The volume model we generate is based on semi-simploidal sets, which guarantees a proper topology and provides a 3D parametric domain for Bézier spaces. This volume is a continuous piecewise Bézier representation which boundary corresponds to a B-Rep made of tensor product Bézier patches.

Keywords: Tubular objects, Topological based geometric modeling, Bézier volumes, Semi-simploidal sets

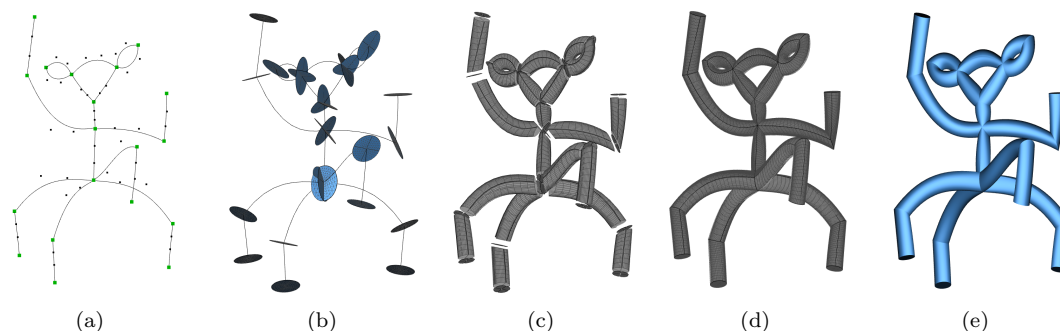


Figure 1: **A parametric volume model from a stick figure.** (a) input: a stick figure, a connected set of Bézier curves; (b) Volume junctions computed at each skeleton vertex; (c) volume branches; (d) a embedded semi-simploidal set defines a topologically coherent parametric domain for volume Bézier patches. (e) output: a piecewise smooth parametric volume model.

1. Introduction

Different geometric models are used to represent 3D objects. In particular, Boundary-representations (B-rep) consists in representing a volume object by its outer surface. The parametric surface models based on Bézier, B-splines or NURBS are the standard models used in the kernel of CAD systems (e.g. Farin (2014)). They offer the possibility of studying and ensuring the required smoothness of the 3D geometric surfaces while providing an intuitive control and easy sampling when needed. Their piecewise linear version,

Email addresses: samuel.peltier@univ-poitiers.fr (Samuel Peltier), geraldine.morin@irit.fr (Géraldine Morin), damien.aholou.pro@gmail.com (Damien Aholou)

7 meshes, have been the most classical representation since they are the natural input of rendering pipelines
8 and the support of texture mapping. They are used in applications where smoothness is not required (e.g.
9 games, e-commerce). More recently, IGA (isogeometric analysis) (see [Cottrell et al. \(2009\)](#) and [Hughes
10 et al. \(2005\)](#)) have proposed to use the geometric model directly for CAM (Computer Aided Manufacturing)
11 based on finite elements computations. With the development of additive manufacturing (3D printing)
12 volume models are needed for representing the geometry of the printed object. When physical properties of
13 a volume object have to be controlled and analyzed (e.g. in the context of additive manufacturing, or 3D
14 printing), a complete volume subdivision has to be handled (i.e. with volume cells) (see [Livesu et al. \(2017\)](#)).
15 Thus, methods for representing the entire volume of an object are in active demand and research. Discrete
16 methods generating volume meshes have been proposed (e.g. [Lyon et al. \(2016\)](#) extract a volume mesh given
17 its surface). [Massarwi and Elber \(2016\)](#) propose a continuous and smooth parametric volume representation
18 based on B-splines; they consider tensor product volume and rely on trimming. These differentiable volume
19 models may be used as support for mechanical computations (see [Massarwi et al. \(2019\)](#)).

20 A skeleton offers an intuitive, simplified and low-dimensional representation of a 3D object structure. It is
21 thus of interest for shape retrieval (see e.g. [Sundar et al. \(2003\)](#)), shape design in sketching and for animation
22 (see e.g. [Raptis et al. \(2011\)](#)). Bridging the gap between skeletons and corresponding surface models has been
23 targeted by different work [Bærentzen et al. \(2012\)](#) and more recently [Usai et al. \(2016\)](#); [Panotopoulou et al.
24 \(2018\)](#); [Fuentes Suárez and Hubert \(2018\)](#). [Livesu et al. \(2016\)](#) generalize these approaches for computing
25 a volume hexahera mesh from a skeleton.

26 The goal of this work is to tackle the limitations of the previous methods, by providing a complete
27 framework for generating automatically a piecewise polynomial parametric tubular volume object from an
28 arbitrary skeleton. Compared to [Livesu et al. \(2017\)](#) that generates a piecewise linear model (3D mesh)
29 for a skeleton, we generate a smooth model based on polynomial elements. Moreover, whereas [Massarwi
30 et al. \(2019\)](#) use volume trimming to propose a smooth volume model, we ensure a consistent topology
31 of the domain so that our model is an assembly of well connected complete volume Bézier patches. More
32 precisely, we associate each edge of the skeleton with a volume *branch*, and each vertex of the skeleton with
33 a *branch junction*. The generated tubular volume object is then defined as an assembly of branches that are
34 glued together continuously at branch junctions. Our work generalizes and extends to **smooth parametric
35 volumes** the results presented in [Panotopoulou et al. \(2018\)](#) that creates a surface quad mesh. In their
36 work, all branches of the surface have the same regular structure: a branch surface is only made of four
37 quads. A similar regularity is provided by the proposed volume model: each volume branch has a regular
38 structure. More precisely, each branch is an assembly of four prisms which boundary is also made of four
39 quads (the four prisms are arranged symmetrically around the branch axis). Note that our method offers the
40 possibility to handle branches made of four hexahera, preserving regularity but generating twice as many
41 quads on the surface.

42 In order to create a parametric model, we use semi-simploidal sets (see [Peltier et al. \(2009\)](#)): a combi-
43 natorial structure in which cells (simploids) are products of simplices; in particular, prisms and hexahera
44 are 3D simploids. Adjacency relations are encoded by face operators, and their structure is designed for
45 embedding semi-simploidal sets into simploidal Bézier spaces (i.e. products of simplicial Bézier spaces).

46 In this paper, we tackle both topological and geometric issues regarding branches, and branch junctions.
47 The main contributions in this paper are the following:

- 48 • Regarding the topology:
 - 49 – We provide definitions of branches and kites (branch extremities) based on semi-simploidal sets.
50 Our branch definition uses prisms and preserves the minimal number of quad boundary patches
51 (i.e. four). Moreover, we extend this property for Bézier volumes model: all branches have a
52 same volume structure and are bounded by only four quads.
 - 53 – The topology of a junction defines how branches are glued together "around" a corresponding ver-
54 tex of the skeleton. The problem of handling junction configurations is equivalent to the problem
55 of partitioning a unit sphere into quads. We provide a proof showing that any quadrangulation
56 of a sphere can be generated iteratively by inserting quads.

- Regarding the geometry:

- During the construction of a quadrangulation on a sphere defining the topology of a junction, we propose an incremental algorithm for positioning automatically the quads vertices. Subsequently, all Bézier volume control points at the junctions are placed.
- We describe how to control the geometry of branches in order to both minimize twisting and preserve a consistent volume along the branch axis. This step places the intermediate Bézier volume control points.

The two following sections of the paper construct the building blocks of the model: the volume branches, and the junctions. Section 2 presents stick figures, the notions and underlying topological structures related to tubular volume branches are detailed. Then Section 3 is dedicated to the topology of junctions. A general iterative process enables the construction of arbitrary junctions. Section 4 describes the complete 3D model construction and focuses on geometric issues regarding junctions and branches. Finally, we show some volume models generated from stick figures with our automatic process.

2. From a Stick Figure to a Volume Tubular Object

In this section, we first introduce the building blocks of the tubular object and discuss how they are intuitively built from a stick figure. The topological nature of tubular volume model is the combinatorial structure of semi-simploidal sets. We recall the basics on this combinatorial structure, and show its benefits for handling both topology and geometry of stick figures and tubular volume objects.

2.1. Building blocks of the tubular object: Stick Figure, Branches and Kites

In this paper, a *stick figure* denotes a skeleton embedded into 3D. More precisely, the structure of a stick figure is a *pseudograph*, i.e. it may contain multi-edges or loops. The embedding of a stick figure is an assembly of Bézier curves: each vertex of the pseudo graph is associated with a 3D point, and each edge is associated with a Bézier curve of arbitrary degree.

The idea behind creating the tubular volume object around the stick figure is to thicken the 1D structure into a volume model. Regarding the structure, an edge of the stick figure corresponds to an elementary volume object, denoted as a *branch*. Regarding the embedding, each Bézier curve C of the stick figure is associated with a smooth (volume) branch whose central curve corresponds to C , and each vertex of the stick figure corresponds to a *junction* of branches. Otherwise said, our tubular objects are assemblies of smooth branches that are glued along their extremities.

Each branch is an assembly of 4 prisms sharing a common edge (Figure 2(a)). Each branch is thus bounded by 4 smooth square surface faces and has two extremities made of an assembly of 4 triangles, denoted as a *kite*. Branches are then assembled along kites according to the algorithm presented in Section 4. In our setting, at a kite a branch is incident to up to 4 other branches.

So, the topological structure of a branch consists in an assembly of 4 prisms, 8 square faces (4 are inside, 4 are on the boundary), 8 triangle face, 21 edges, and 10 vertices. All these cells correspond to a same kind of regular cells: simploids [Dahmen and Micchelli \(1982\)](#). Intuitively, a simploid can be seen as the product of (any) simplices. Assemblies of simploids can be handled using semi-simploidal sets: a combinatorial structure for handling simploidal Bézier spaces. More precisely, simploids are a natural domain for Bézier volumes, and so, represent non linear, polynomial models.

Note that, a volume branch could also be defined by a set of four hexahedra and benefit from a semi-simploidal set representation, leading to a coherent topology and a (tensor-product) Bézier embedding, as shown in Figure 2(b). In order to keep the property of having a minimal number of surface patches shown in [Panotopoulou et al. \(2018\)](#), we thus chose to define branches as assemblies of prisms. However, despite having twice as many surface patches (i.e. eight), all our work and proposal does directly carry through with 4-hexahedra branches.

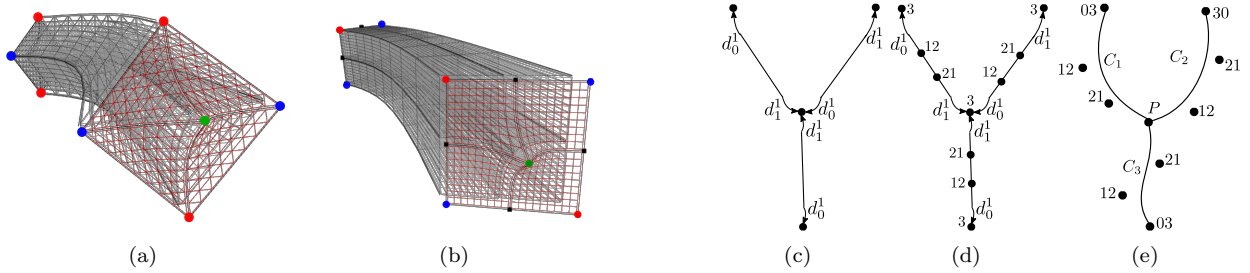


Figure 2: (a) A volume branch which domains are prisms has 4 quad patches on its outer boundary. (b) A volume branch which domains are hexahedra has 8 quad patches on its outer boundary. (c) – (e) an example of a stick figure represented as an assembly of 3 Bézier curves sharing a common vertex. (c) its topology is encoded by a semi-simploidal set of dimension 1. (d) numbering of control points induced by face operators. Note that the control point P is common to the three curves, and its numbering on each curve depends on the face operators: P is numbered 30 for C_1 and C_3 as it corresponds to the face d_1^1 of C_1 and C_3 , whereas P is numbered 03 for C_2 as it corresponds to the face d_0^1 of C_2 . Note that Bézier curves have the same degree in this example, but different degrees can be handled.

2.2. Semi-simploidal Sets

The topological structure of our proposed volume model is based on semi-simploidal sets. This combinatorial structure is designed to define assemblies of Bézier volume patches and is the building block of the proposed volume model. In this section, we recall their definition (Section 2.2.1) and their Bézier embedding (Section 2.2.2).

Abstract simplicial complexes Munkres (1984) are the most well-known structure for representing simplicial objects. Within this structure, simplices are defined as sets of vertices. Semi-simplicial sets or Δ -complexes Eilenberg and Zilber (1950); Hatcher (2002) also describe simplicial objects, but in these structures, simplices are defined as sequences of vertices; such structures allow in particular to handle simplicial objects with multi-incidence (for example, an edge loop or two triangles sharing their 3 edges). Semi-simplicial sets are defined as a set of abstract cells (simplices) together with face operators satisfying some properties that ensure a consistent topology. Structural relations have been established between face operators and control points of triangular Bézier spaces Lang and Lienhardt (1996), leading to the definition of data structures allowing to handle the topology of free-form objects, while benefiting from a built-in parametric domain.

In a similar way, semi-cubical sets Brown and Higgins (1981) have been defined for handling assemblies of cubes (i.e. products of edges). More recently, semi-simploidal sets Peltier et al. (2009) have been defined in a similar way for handling assemblies of Bézier simploids DeRose et al. (1993). Intuitively, a simploid Dahmen and Micchelli (1982) corresponds to a product of any simplices, so simplices and cubes are particular simploids.

In the following, we recall the definition of semi-simploidal sets, and their relations to corresponding Bézier spaces (see Peltier and Lienhardt (2018) for theory and implementation details). We then define stick figures, branches and kites using semi-simploidal sets.

2.2.1. Combinatorial structure

A semi-simploidal set is a combinatorial structure, defined as a set of simploids equipped with face operators. A simploid σ is characterized by its type (a_1, \dots, a_n) , where a_i corresponds to the dimension of its i^{th} generating simplex, and $a_i > 0$ for any $1 \leq i \leq n$ (intuitively, the cartesian product of a simplex σ by a vertex, i.e. a simplex of dimension 0, is σ). The dimension k of σ is $\sum_{i=1}^n a_i$, i.e., the sum of the dimensions of its generating simplices, and the face operators associates σ with its faces (i.e. simploids of dimension $k - 1$). For example, a square is a simploid of type $(1, 1)$, thus dimension 2, as it corresponds to the product of two edges, its faces are simploids of type (1) i.e. edges. A prism is a simploid of type $(2, 1)$, thus dimension 3, as it corresponds to the product of a triangle by an edge. The faces of a prism are simploids type (2) (i.e. triangles) and $(1, 1)$ (i.e. squares), both have dimension 2. Note that a simplex and a cube of dimension k are particular simploid of type (k) and $(1, \dots, 1)$ with length k respectively.

137 **Definition 1.** A semi-simploidal set $S = (K, (d_j^i))$ is a set of simploids K equipped with a type operator
138 $\mathcal{T} : K \rightarrow \bigcup_{i=0}^{\infty} \mathbb{N}^{*i}$ and face operators d_j^i . Let $\sigma \in K$; $\sigma\mathcal{T}$ is the type of σ . Let $\sigma\mathcal{T} = (a_1, \dots, a_n)$: σd_j^i is
139 defined if $1 \leq i \leq n, 0 \leq j \leq a_i$. Operators satisfy:

140

(I) Action on the type

(II) Commutation of face operators

141

$$\sigma d_j^i \mathcal{T} = \begin{cases} (a_1, \dots, a_i - 1, \dots, a_n) & \text{if } a_i > 1 \\ (a_1, \dots, a_{i-1}, a_{i+1}, \dots, a_n) & \text{otherwise} \end{cases}$$

$$d_j^i d_l^i = d_l^i d_{j-1}^i \quad l < j, a_i > 1$$

$$d_j^i d_l^k = \begin{cases} d_l^k d_j^i & \text{if } a_k > 1 \\ d_l^k d_{j-1}^i & \text{otherwise} \end{cases} \quad k < i$$

142 The face operator d_j^i associates with σ the same product of simplices, except that the j^{th} generating
143 simplex is replaced by its j^{th} face. So, the type of σd_j^i is $(a_1, \dots, a_i - 1, \dots, a_n)$ if this j^{th} face is not a
144 vertex, $(a_1, \dots, a_{i-1}, a_{i+1}, \dots, a_n)$ otherwise. Note that σd_j^i denotes $d_j^i(\sigma)$.

145 In this work, stick figures (i.e. graphs), are handled by semi-simploidal sets of dimension 1 (cf. Figure
146 2(c) – (e)). For the volume model, only one type of simploid of dimension 3 is used: prisms. Let σ be
147 a prism, then σ is equipped with 5 boundary operators: d_0^1, d_1^1, d_2^1 which point to its three faces of type
148 (1, 1) (i.e. square faces) and d_0^2, d_1^2 which points to its two faces of type (2) (i.e. triangles). Commutation
149 properties guarantee a consistent topology. For example, the square face σd_1^1 and the triangle face σd_0^2 share
150 the common edge $\sigma d_1^1 d_0^2 = \sigma d_0^2 d_1^1$ (cf. Figure 3(b)).

151 2.2.2. Bézier Embedding

152 In addition to a consistent topological structure, semi-simploidal sets also offer a domain for Bézier
153 volume shape. This section briefly recalls the general setting, and highlights the specific Bézier elements the
154 branches are made of.

155 A simploidal Bézier space is a set S of Bézier simploids such that the intersection of any two simploids
156 σ and τ of S is either empty or a simploid of S , corresponding to a common face of σ and τ .

The set Γ_d^i of i -dimensional multi-indices of degree d is defined by $\Gamma_d^i = \{\alpha = (\alpha_0, \dots, \alpha_i) \in \mathbb{N}^{i+1} \mid |\alpha| = \alpha_0 + \dots + \alpha_i = d\}$. Multivariate Bernstein polynomials of degree d are defined at v by $B_\alpha^d(v) = \binom{d}{\alpha} v_0^{\alpha_0} \dots v_i^{\alpha_i}$, with $\alpha \in \Gamma_d^i$, and $\binom{d}{\alpha} = \frac{d!}{\alpha_0! \dots \alpha_i!}$ are multinomial coefficients, and $\{v_j\}$ are the barycentric coordinates of v , a point of the standard i -simplex, i.e. $\{v_j\}$ satisfies: $\forall j, 0 \leq j \leq i, 0 \leq v_j \leq 1$ and $\sum_{j=0}^i v_j = 1$. A Bézier simploid of type (a_1, \dots, a_n) and degree (d_1, \dots, d_n) is defined by:

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

157 where any u^i is a point of the standard a_i -simplex and $\{P_{(\alpha^1, \dots, \alpha^n)}\}$ is its set of control points.

158 A semi-simploidal set structure can be associated with a simploidal Bézier space, as a direct correspon-
159 dence exists between a semi-simploidal set and the structure of its control points. This correspondence is
160 based on the fact that for any control points of a Bézier simploid σ having a zero on a given multi-index po-
161 sition, means that it also corresponds to a control point of a Bézier simploid of the boundary of σ . Moreover,
162 the position of the zero corresponds to the index of the face operator.

163 In our setting (see Figure 3 (c) – (d)), we consider prism μ of type (2, 1) and degree (3, 3). The only
164 control points that are inside μ are $P_{(111,12)}$ and $P_{(111,21)}$; the sets of control points $\{P_{(\alpha^1, 03)}\}$ and $\{P_{(\alpha^1, 30)}\}$
165 correspond to their triangular face μd_0^2 and μd_1^2 respectively.

166 2.3. Semi-simploidal Branches and Kites

167 As mentioned before, branches are built by "gluing" four prisms together as illustrated in Figure 2(a).
168 Gluing corresponds more formally to an identification operation (see Peltier et al. (2009) Section 3.2.2.
169 for details). Identifying two simploids and their boundary consists in merging them into a single simploid,
170 according to their orientation (induced by face operators). Regarding branches, each prism shares a common
171 quad face with each of its two neighbors. The identification operation will be also used when branches

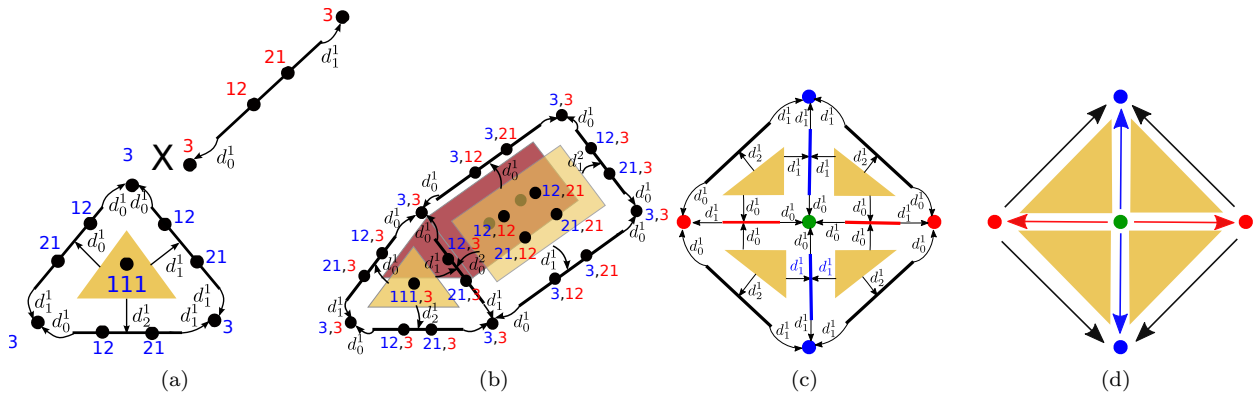


Figure 3: (a) – (b): Correspondences between control points of a Bézier prism and its face operators represented by arrows. The prism μ on (b) (and its faces) is the product of the triangle σ_1 (and its faces) and the edge σ_2 (and its faces) on (a). The square face represented on (b) is $\mu d_1^1 = \sigma_1 d_1^1 \times \sigma_2$; the bottom square face of μ is $\mu d_2^1 = \sigma_1 d_2^1 \times \sigma_2$; the square face on the back side is $\mu d_0^1 = \sigma_1 d_0^1 \times \sigma_2$; the front triangle face is $\mu d_0^2 = \sigma_1 \times \sigma_2 d_0^1$; the back triangle face is $\mu d_1^2 = \sigma_1 \times \sigma_2 d_1^1$. The prism contains only two proper control points of the volume $P_{111,12}$ and $P_{111,21}$; all other control points are also control points of a Bézier simploid of dimension 2, 1 or 0. (c) a kite, the extremity of a branch, is a semi-simploidal set of dimension 2. This specific assembly of 4 prisms leads to symmetries on the kite. (d) simploid coloration is sufficient to recover the face operators of a kite.

172 are glued along part of their kites. Topological correctness of each branch ensures the correctness of the
 173 corresponding Bézier structures. As a consequence, two incident Bézier prisms share a common Bézier quad
 174 surface, and the central Bézier curve is a single curve object shared by the four Bézier prisms and the four
 175 inside Bézier quad surfaces. Note that Bézier prisms are defined by volume control points.

176 On top of the four outer surface boundary quad, a branch now has two extremities, called *kites*. Figure
 177 3(c) illustrates the simploidal structure of these kites. Our specific assembly of 4 prisms ensures that kites
 178 have an interesting property of axial symmetry regarding simploid orientations (see. Figure 3(d)). Thus, to
 179 each vertex of the stick figure corresponds a set of kites in the volume object, assembled into a so called
 180 *kite folio*.

181 3. Handling Topology of Junctions

182 In this part, we provide all the material for generating *any* branch junction corresponding to a vertex
 183 of a stick figure, through an iterative process. Whereas this section concentrates on the arrangement of
 184 branches, filling in the corresponding volume is addressed in Section 4.

185 We consider a unit sphere around each vertex v of degree $n \geq 2$ of the stick figure: each edge incident to
 186 v is embedded as a Bézier curve, and its tangent at the vertex defines an exit ray which meets the sphere
 187 in one point; we denote these points *green points*. A quad tessellation (or quadrangulation) of this sphere
 188 is built incrementally while considering branches at this junction. Then each branch extremity is plugged
 189 onto its corresponding quad in the quadrangulation.

190 3.1. Simple Quadrangulations on a Sphere: Properties and Orientation

191 In this paper, we consider *simple quadrangulations* on a sphere (or SQS for short): a class of simple planar
 192 graphs (i.e. with no self loop nor double edges) such that each face is bounded by four vertices. Simple
 193 quadrangulations on surfaces have been studied in Nakamoto (1996). Batagelj (1989) studies classes of
 194 quadrangulations generated by inductive definitions, i.e. starting from an initial configuration and generating
 195 rules (adding quads). Following this approach, Brinkmann et al. (2005) defines *expansions* operations and
 196 their inverse, *face reductions* involving only vertices of minimal degree 2 or 3. In fact he defines special case
 197 of the general *face contraction* operation defined in Nakamoto (1996). Roughly speaking, these operations
 198 consist in adding or removing a quad while preserving a quadrangulation. He shows that any SQS can

199 be reduced to the minimal one, made of only two faces, by a sequence of face reductions applied to faces
200 incident to a vertex of minimal degree.

201 [Panotopoulou et al. \(2018\)](#) also investigate the problem of generating SQS. Their approach is based
202 on splitting quads, which corresponds one expansion operation of [Brinkmann et al. \(2005\)](#). Indeed, in the
203 resulting quadrangulation, at least 2 quads share 2 adjacent edges, so at least two branches are twice incident
204 to each other. As a consequence, configurations where each branch has four different neighbors can not be
205 obtained, e.g. the surface of a dice involving six branches (see Figure 5 (d) – (f)). Such configurations are
206 important, in particular, when the six branches join in the six canonical, orthogonal directions.

207 In our context, where expanding and reducing a quad corresponds to adding or removing a branch of
208 the volume object, it is essential to be able to chose the location of the quad. In the following, we provide
209 all the necessary properties to prove Theorem 1. In terms of graph generation our result is similar to the
210 one of [Brinkmann et al. \(2005\)](#), but the proof we provide is more relevant for geometric consideration since
211 it is based on the fact that in an SQS with $n > 2$ quads, any chosen quad can be reduced (Proposition 6).

212 *Properties of SQS*

213 Note that a SQS is bipartite, and the length of any cycle in a SQS is even, and is greater or equal to 4.

214 **Property 1.** *Let Q be a SQS having $n \geq 2$ quads. If $n = 2$, then the two quads share their 4 edges, else*
215 *two quads of Q do not share opposite edges.*

216 *Proof.* Let us consider a SQS having $n > 2$ faces, and F_1, F_2 be two of its faces. Let us assume that F_1 and
217 F_2 share 2 opposite edges. Thus F_1 and F_2 share the same 4 vertices. As F_1 and F_2 do not share their 4
218 edges, there is at least one double edge, which is a contradiction with the definition of a SQS (which contains
219 no multi-edge). □

220 As a direct consequence, the following properties hold:

221 **Property 2.** *In a SQS having $n > 2$ quads, any two quads share at most 2 (adjacent) edges.*

222 **Property 3.** *Let (a, b, c, d) be a quad of a SQS such that b and d have a neighbor vertex in common*
223 *(different from a and c). Then, a and c do not have any neighbor vertex in common (different from b and*
224 *d).*

225 *Proof.* Let us assume that both a and c have a common neighbor vertex v , and b and d have a common
226 neighbor vertex w .

- 227 • Let us assume that $v = w$. A SQS is bipartite, so a, c and w have the same color and b, d and v the
228 other color. So v and w have different colors, so v can not be equal to w , which leads to a contradiction.
- 229 • Let us assume $v \neq w$. The boundary of the quad (a, b, c, d) is by definition also the boundary of the
230 rest of the SQS. Any two paths linking a to c in one hand, and b and d on another hand must intersect
231 as they are opposite vertices of a quad, but the only vertex in these two paths are v and w , and $v \neq w$
232 which leads to a contradiction.

233 □

234
235 **Property 4.** *Let Q be a SQS with $n > 2$ quads, and let (a, b, c, d) be a quad of Q such that b and d do not*
236 *have a neighbor vertex in common (different from a and c). Then, a and c have at least degree 3.*

237 *Proof.* If b or d has degree 2, then property 2 ensures that a and c have at least degree 3. If b and d have
238 at least degree 3, then let us assume that a has degree 2, then the second face containing a also contains b
239 and d , thus b and d must have a common vertex. Which leads to a contradiction. The same result holds for
240 d . □

241 In practice, SQS vertices, edges and faces (here *quads*) are handled through the combinatorial structure
 242 of oriented maps which provides orientation of cells of the subdivision (also used in Section 4). In the
 243 following, we define expansion and reduction of quads for oriented maps, namely *opening* and *closing*.

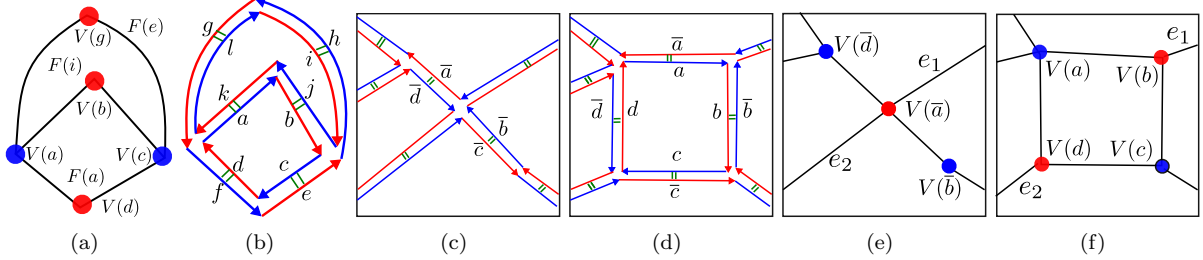


Figure 4: (a) : a SQS with 3 faces, 6 edges, and 5 vertices. (b) : its corresponding oriented map with 12 darts labelled from a to l . Each dart is represented by an arrow, which direction encodes permutation β_1 ; involutions β_2 are represented in green between two opposite darts. (c) – (d): Focus on a quad. From (c) to (d): opening operation relatively to darts \bar{a} and \bar{c} . From (d) to (c): closing operation relatively to dart a (or c). SP(e), (f): the SQS corresponding to (c) and (d) respectively. Note that after opening: $V(\bar{d}) = V(a)$ and $V(\bar{b}) = V(c)$.

244 Oriented Maps

245 Oriented maps are designed to represent any subdivision (not necessarily quad faces) of orientable surfaces
 246 without boundaries. The concept was first introduced by Edmonds (1960), and is also known as *half-edge*
 247 (see. Weiler (1985); Mäntylä (1988)) ; it has been extended for orientable or not orientable surfaces by
 248 Guibas and Stolfi (1985), to dimension 3 by Dobkin and Laszlo (1987), to any dimension by Vince (1983);
 249 Lienhardt (1989). Equivalence between these structures has been shown in Lienhardt (1991). Interested
 250 readers can find more details about such structures in Damiand and Lienhardt (2014).

251 **Definition 2.** An oriented map is a triplet (H, β_1, β_2) where H is a finite set of elements called darts, β_1
 252 is a permutation on H , and β_2 is an involution on H .

253 Intuitively, within this structure, each edge is decomposed into two opposite darts (also called half-edges).
 254 Each dart h is associated with a unique vertex, edge, and face denoted $V(h)$, $E(h)$ and $F(h)$ respectively.
 255 All vertices, edges and faces correspond to cycles $\beta_2\beta_1$, β_2 and β_1 , respectively. In other words, 2 opposite
 256 darts corresponding to the same edge are linked by involution β_2 , while the permutation β_1 allows to traverse
 257 the boundary of a face, and permutation $\beta_2\beta_1$ allows to turn around each vertex. For example, on Figure
 258 4(b), face $F(a) = F(b) = F(c) = F(d)$ is defined by the permutation (a, b, c, d) ; edge $E(a) = E(k)$ by the
 259 permutation (a, k) ; vertex $V(c) = V(h) = V(j)$ by the permutation (c, h, j) . By definition, each face of the
 260 map corresponds to an elementary cycle in the corresponding SQS Q .

261 Since Q is bipartite, we can color its vertices blue and red. Note that each dart is associated to a unique
 262 vertex, hence a red or blue color can be associated to each dart, according to its corresponding vertex as
 263 shown in Figures 4(a) and 4(b).

264 3.2. Opening and Closing operations

265 In the following, *SQS oriented map* denotes an oriented map corresponding to a SQS. Figure 4 illustrates
 266 an opening (Definition 3, from c to d) and a closing (Definition 4 from d to c).

267 **Definition 3.** Let $Q' = (H', \beta'_1, \beta'_2)$ a SQS oriented map, and $\bar{a}, \bar{c} \in H'$ s.t. $V(\bar{a}) = V(\bar{c})$. Let $\bar{d} = \bar{a}\beta'_2$ and
 268 $\bar{b} = \bar{c}\beta'_2$, the **opening** (Q', \bar{a}, \bar{c}) is the SQS oriented map $Q = (H, \beta_1, \beta_2)$ defined by :

- $H = H' \cup \{a, b, c, d\}$
- $\forall h \in H', h\beta_1 = h\beta'_1;$
- $a\beta_1 = b; b\beta_1 = c; c\beta_1 = d; d\beta_1 = a$, that is, (a, b, c, d) is the new quad;
- $\forall h \in H' - \{\bar{a}, \bar{b}, \bar{c}, \bar{d}\}, h\beta_2 = h\beta'_2;$
- $\bar{a}\beta_2 = a; a\beta_2 = \bar{a}; \bar{b}\beta_2 = b; b\beta_2 = \bar{b};$
 $\bar{c}\beta_2 = c; c\beta_2 = \bar{c}; \bar{d}\beta_2 = d; d\beta_2 = \bar{d}.$

270 **Definition 4.** Let $Q = (H, \beta_1, \beta_2)$ a SQS oriented map with $n > 2$ quads, and (a, b, c, d) a quad of Q such
 271 that $V(b)$ and $V(d)$ do not have a common neighbor vertex other than $V(a)$ and $V(c)$. We denote $\bar{a} = a\beta_2$,
 272 $\bar{b} = b\beta_2$, $\bar{c} = c\beta_2$ and $\bar{d} = d\beta_2$. The **closing** (Q, a) is the SQS oriented map $Q' = (H', \beta'_1, \beta'_2)$ defined by:

- 273 • $H' = H - \{a, b, c, d\}$ • $\forall h \in H' - \{\bar{a}, \bar{b}, \bar{c}, \bar{d}\}, h\beta'_2 = h\beta_2;$
- 274 • $\forall h \in H', h\beta'_1 = h\beta_1;$ • $\bar{a}\beta'_2 = \bar{d} ; \bar{d}\beta'_2 = \bar{a} ; \bar{b}\beta'_2 = \bar{c} ; \bar{c}\beta'_2 = \bar{b}.$

275 Note that the *opening* operation is the inverse of the *closing* operation. This can easily be checked by
 276 applying directly the definitions. Note also that $\text{closing}(Q, a)$ is equivalent to $\text{closing}(Q, c)$. An important
 277 result (Property 6) is that any quad (a, b, c, d) can be closed along at least one of its two diagonals (a, c) or
 278 (b, d) .

279 **Property 5.** Let $Q = (H, \beta_1, \beta_2)$ be a SQS oriented map, $a, c \in H$ and $Q' = \text{closing}(Q, a)$. Then Q' is a
 280 SQS oriented map.

- 281 *Proof.* 1. In Definition 4, the precondition ensures that $V(b)$ and $V(d)$ do not have any neighbor vertex in
 282 common. So, Property 4 ensures that $V(a)$ and $V(c)$ have both degree at least 3. So the corresponding
 283 vertices in Q' have same degree minus one, which remains ≥ 2 . Moreover, $V(b)$ and $V(d)$ are merged
 284 in Q' into a single one having degree $\text{deg}(V(b)) + \text{deg}(V(d)) - 2$, which is by definition also ≥ 2 . So
 285 **all vertices of Q' have degrees greater or equal to 2.**
- 286 2. All the faces of an oriented map are cycles β_1 . According to Definition 4, no β_1 is modified, so all
 287 faces keep the same number of edges in their boundary after the operation. As Q' is obtained from a
 288 SQS Q , **all the faces of Q' remain quads.**
- 289 3. By construction, the two vertices that are merged are opposite points (same color) of a quad, so **no**
 290 **loop can appear after a closing.**
- 291 4. The only multi-edge that could be created are the one incident to $V(b)$ and $V(d)$ before the operation.
 292 This would be the case if these two vertices share a common neighbor vertex, but the precondition
 293 states that they do not share a common neighbor vertex. So Q' **contains no multi edge.**

294 □

295 3.3. Generating any SQS with a sequence of openings

296 The closing operation has been defined in order to prove Theorem 1.

297 **Property 6.** In a SQS having $n > 2$ quads, any quad can be closed.

298 *Proof.* The proof is a direct consequence of Property 3 and Property 4. □

299 **Property 7.** The SQS having $n = 2$ quads can be obtained from any SQS by any sequence of closing
 300 operations.

301 *Proof.* Property 6 ensures that for any SQS with $n > 2$ quads, any quad can be closed. As a direct
 302 consequence, given an arbitrary SQS with $n > 2$ quads, $n - 2$ quads can be iteratively chosen and closed to
 303 obtain a SQS with $n = 2$ quads. □

304 We can now establish the following theorem, which ensures that any SQS can be obtained with a sequence
 305 of openings:

306 **Theorem 1.** Any SQS having $n \geq 2$ quads can be obtained from the canonical SQS having 2 quads, by
 307 performing a sequence of $n - 2$ openings.

308 *Proof.* The proof is straightforward using Property 7 and the fact that opening is the inverse of closing. □

309 From a SQS, a surface tubular object can be created by matching each branch boundary to its corre-
 310 sponding quad. In the process of creating a complete volume object, the next section goes one step further
 311 by connecting the volume branches inside the junction spheres, leading to a continuous volume object.

312 4. Creating a Complete Tubular Volume Model and Defining its Geometry

313 In the two previous sections, we have shown how to define the topology of the 3D model from the stick
 314 figure. Namely, Section 2 defines the 3D branches, and Section 3 shows how to generate an SQS for handling
 315 junctions of arbitrary topology: the SQS is built incrementally at each junction, by a sequence of openings.
 316 This sequence corresponds to an ordering on the green points inherited from an incremental construction of
 317 stick figure.

318 In this section, we create the topology of the complete volume model and automatically define its
 319 geometry by positioning of its control points. Section 4.1 describes how the topology of the volume object is
 320 defined: first a kite folio is generated from the SQS, then all branches are plugged into corresponding kites
 321 according to a chosen red and blue coloring. Section 4.2 focuses on the geometry: the first part addresses
 322 junctions and the second one branches.

323 4.1. Topology of the 3D Model

324 As mentioned before, to each vertex of the stick figure corresponds a junction of volume branches. We
 325 denote these volume junctions *kite folio* (cf. Figure 1 (b)). We establish the following process for building
 326 the topology of a tubular object from a stick figure:

- 327 • When adding a curve to the stick figure, each extremity of the corresponding branch is inserted into
 328 SQS, giving the opening support. Once all curves have been processed, an SQS is built around each
 329 vertex (Section 3);
- 330 • Then, from an SQS, a kite folio is computed. This kite folio ensures a continuous junction of the
 331 volume branches, connecting their extremities (kites) within the sphere;
- 332 • Now, each branch needs to be connected to its corresponding kite. Before linking the kite folios by
 333 plugging branches, a traversal of the stick figure propagates a red/blue coloration of junction SQSs.

334 Below, we detail the last two points.

335 *From an SQS to a kite folio*

336 At the end of the construction of an SQS Q (with a green point hosted in each quad), Q can directly be
 337 converted into a semi-simploidal set S of dimension 1 such that each edge $e = (v_{red}, v_{blue})$ in Q is mapped to
 338 an edge σ of S , such that $\sigma d_0^1 = \mu$ and $\sigma d_1^1 = \mu'$ where μ and μ' corresponds to v_{red} and v_{blue} respectively.

339 Then, a kite folio can directly be obtained by applying a *cone* operation¹ on S with a new vertex v ,
 340 which we choose to be the sphere center. More precisely, the kite folio is defined as follows: for each edge
 341 σ (resp. vertex μ) of S , a new triangle $\bar{\sigma}$ (resp. edge $\bar{\mu}$) is added, such that: $\bar{\mu}d_0^1 = v$, $\bar{\mu}d_1^1 = \mu$, $\bar{\sigma}d_2^1 = \bar{\sigma}$,
 342 $\bar{\sigma}d_0^1 = \bar{\sigma}d_0^1$, and $\bar{\sigma}d_1^1 = \bar{\sigma}d_1^1$. Note that this operation creates a semi-simploidal set of dimension 2 such that
 343 all triangles are incident to v . Figure 3(c) illustrates the cone operation on a single quad boundary of Q
 344 with the green central vertex: each internal edge $\bar{\mu}$ comes from a red or blue vertex μ ; its face d_0^1 is the
 345 green vertex, and its face d_1^1 is μ . Similarly, each triangle $\bar{\sigma}$ comes from an edge σ of the boundary quad,
 346 which is its face d_2^1 ; its faces d_0^1 and d_1^1 are the internal edges $\bar{\mu}$ which comes from σd_0^1 and σd_1^1 respectively.
 347 The cone on S with v , the sphere center, results in a kite folio (see Figure 5 (e)): as any edge is shared by
 348 exactly two quads in Q , any triangle of the kite folio is shared by exactly two kites. Otherwise said, each
 349 triangle is the support for a "quarter of" two branches (one on each side).

350 Applying this process to an entire stick figure generates a set of kite folios intuitively connected by the
 351 stick figure curves (see Figure 1 (b)).

¹The general cone operation is not recalled here, (see Peltier et al. (2009), Section 2.2.1. for details.)

352 *Plugging branches into the set of kite folios*

353 As a branch is plugged into its kite through an identification, the vertex colors have to be matched (as
354 explained in Figure 3(c)). So, a red (resp. blue) vertex of the branch kite is matched with one of the two
355 red (resp. blue) vertices of its corresponding kite in the kite folio. It is sufficient to select only one (red or
356 blue) point, identification of the two kites follows directly; thus, this offers two possible kite identifications.

357 Moreover, we have an extra degree of freedom per junction: on a kite folio, red and blue vertex colors can
358 be swapped. Swapping the colors on a kite folio changes the orientation of the four triangles of all kites (as
359 illustrated on a single kite on Figure 3(d)). To determine sequentially the choice of color at each junction,
360 we traverse the stick figure by a BFS (breadth first search) starting at the node of maximum depth (degree
361 one stick graph vertices and loops are considered depth zero) and maximal degree. The color of this first
362 kite folio is arbitrary, but then, a coloring is chosen for each traversed kite folio, according to geometric
363 criteria based on branch orientation detailed in section 4.2.2.

364 *4.2. Geometry of the 3D Model*

365 Now that we have addressed all topological considerations for creating a complete model, this section
366 addresses geometric considerations for embedding the complete model in 3D.

367 A direct user based option for positioning the branches at junctions could directly be implemented: while
368 building a stick figure, the user chooses a sequence of openings for each junction, and sets the position of
369 each red and blue vertex in the corresponding junction. Then, the internal control points of the branch can
370 be also positioned. Such a solution provides entire control of the structure and geometry of each branching,
371 but it may take a lot of interactions to build a volume object corresponding to a stick figure.

372 *4.2.1. Geometry of junctions*

373 Given the sequence of openings that generates an SQS Q and then a kite folio, we now explain the
374 heuristic we propose for positioning red and blue vertices of Q at each step. To achieve this goal, we rely
375 on Q^* the dual of Q , whose vertices are the green points, and faces correspond to red and blue points.

376 *Dual of an SQS*

377 Let Q be an SQS, we define the *pinching* operation in Q^* , corresponding to an *opening* in Q (see Figure
378 5). The motivation for considering Q^* , the dual of Q , comes from the fact that the positions of the (green)
379 vertices are determined by the stick graph. In Section 4.2.1, given those green points, we determine where
380 to place the red and blue vertices of Q .

381 Note that Q^* is a planar graph, like Q , but is not an SQS. In practice, Q^* is also represented by an
382 oriented map. Q^* has the following properties:

- 383 • The faces of Q^* inherit an orientation from Q ;
- 384 • Q^* is not simple: when two quads of Q share two incident edges, Q^* has a double edge, thus a 2-sided
385 face. Note that Q^* can not have any loop since Q has no degree 1 vertex;
- 386 • Q^* is 4-regular: the faces of Q are quads, so each green vertex of Q^* is incident to exactly four edges.

387 Junctions are built using a sequence of openings: in practice, the corresponding dual pinching operations
388 are processed in the dual oriented map representing Q^* .

389 *Choice of pinching*

390 At each step of an SQS construction, a new edge from the stick figure is considered, thus a new green
391 vertex g is added on the sphere. This new vertex g belongs to exactly one face f of Q^* . Considering the
392 position of g (which is fixed by the stick figure) in f , a pinching operation is determined by the choice of
393 two edges of f . Intuitively, these chosen edges are each split into two edges adjacent to g , splitting the face
394 f into two faces (Figure 5 (a) – (c)). The choice of the two edges to pinch is based on two criteria:

- a connectivity criterion: in order to balance the number of edges in the resulting faces, we select the edges to pinch on opposite sides of the face. Note that choosing two adjacent edges would create a new double edge in Q^* .
- a geometry criterion: in an attempt to create faces with comparable surface areas, we select edges that update faces so that their change in surface area after pinching is the most comparable.

In practice, it is unlikely to select a face f bounded by a double edge unless the new vertex g lies exactly on the geodesic line between its 2 vertices. However, the heuristic we propose tackles this issue: once f (with 3 or more vertices) is determined, we check if any edge e of f is a double edge. Then, if it is the case, and if e is closer to g than the center of f , the 2-sided face bordered by e is selected for the pinching instead of f .

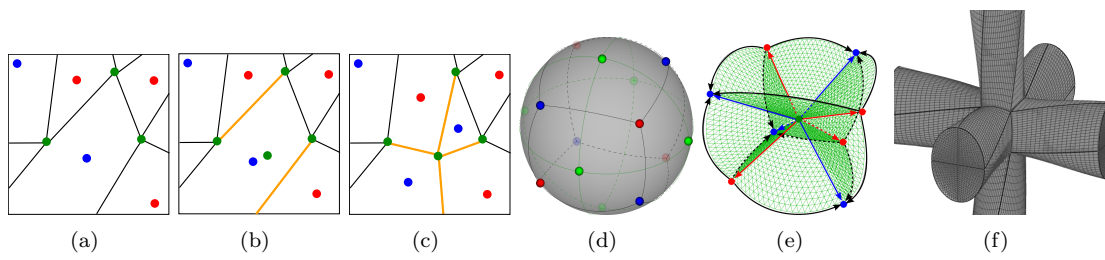


Figure 5: (a) – (c): **pinching on Q^*** of two edges of a face corresponds to an opening on Q . (a): Q^* with n green points. (b) adding a new green vertex on the face of a blue point and choosing the two edges in Q^* to be pinched (in orange). (c) result of the pinching creates a new face in Q^* . (d) – (e): **The dice configuration**. (d): an SQS (red and blue points) Q in "dice" configuration together with its dual (green points). (e) applying a cone operation on the 1D semi-simploidal set S obtained from Q with the sphere center generate a kite folio. (f): the same dice configuration with volume branches.

405 *Placing the red and blue vertices*

406 Now that Q^* is entirely defined, we describe how red and blue vertices of Q are placed on the sphere
 407 during the sequence of pinchings. Remember that red and blue vertices of Q corresponds to faces in Q^* ,
 408 and faces in Q^* are bounded by 2, 3 or more green vertices. At the beginning of the process, we consider
 409 first two green points, thus the corresponding SQS corresponds to the base case of two branches, where the
 410 4 points (2 blue, 2 red) are distributed on the plane bisector of the two green points (such junction with
 411 two branches appears on Figure 1 (b)). Then, for junctions with more than 2 branches, for each new green
 412 vertex that is added, we place the red and blue vertices on the sphere, at a so-called *center* of a face of Q^*
 413 depending on its number of vertices:

- For faces bounded by two green vertices (those are bounded by a double edge of Q^*): the *center* is the midpoint of the arc in the oriented face between the two green vertices.
- For faces bounded by three green vertices or more, we compute the least square plane to all vertices of the face. When a face contains an edge corresponding to a geodesic longer than π , then the middle point of this geodesic is also taken into account for computing the plane. This plane intersects the sphere in a circle. We choose the projection so that the *center* is inside the face according to the orientation given by the oriented map. The least square plane is computed using the Moore-Penrose pseudoinverse. In practice, we use the linear algebra software library Armadillo (see Sanderson and Curtin (2016)).

423 Note that for faces bounded by three green vertices, the *center* is the projection of the circumcenter of
 424 their triangle on the sphere, which is equidistant from all three green vertices. For more vertices, the center
 425 is thus equidistant, in a least square sense, to the vertices on the boundary of the face. Each quad face
 426 contains exactly a branch. When branches are from nearby directions, the topology remains coherent and

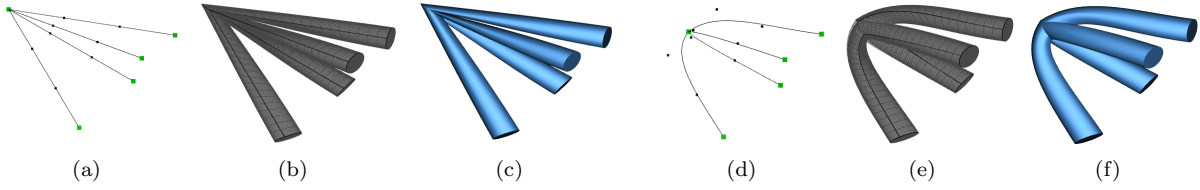


Figure 6: (a) – (c) A junction with four branches from nearby directions. Note that there is exactly one quad face per branch and the topology remains coherent. (d) – (f) By changing the tangent of the branches, the junction is smoothed.

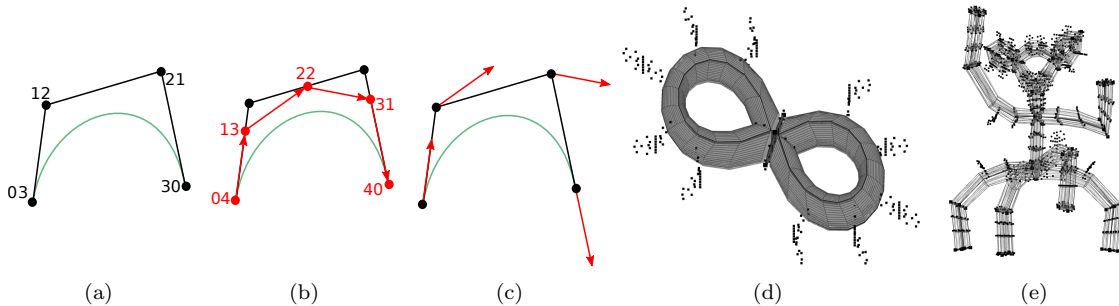


Figure 7: (a) – (c): The control tangents (in red) at each Bézier control points are computed from the segment of the degree elevated control polygon (b). (d) – (e): Placement of volume control points of branches.

427 the junction has a sharp corner. As our model offers curve branches, sharp corners can be simply avoided
 428 by smoothing the tangent direction at the junction as illustrated in Figure 6.

429
 430 The order in which edges are processed at a junction affects the resulting SQS thus the kite folio that
 431 is obtained, regarding both its topology and its geometry. So, the resulting tubular object depends on this
 432 ordering. In practice, the user that is drawing a stick figure implicitly creates a total order on the edges to
 433 consider at each junction; green points are inserted in this order.

434 4.2.2. Geometry of branches

435 Each branch is attached to junctions through the kites at its extremities. We detail here the geometry
 436 criteria to ensure the best relative orientation of these two extremities. This geometric consideration com-
 437 pletes the topological approach detailed in Section 4.1. Then, the control points of branch extremities are
 438 placed into the plane of each corresponding kite triangle. We now need to control what happens within the
 439 branch volume by defining Bézier elements between the two extremity kites. The central edge of a branch
 440 corresponds exactly to the edge in the stick figure, we then explain how we position the remaining internal
 441 control points of the branch. Two main considerations improve the geometric aspect of our branches:

- 442 • **Minimize twisting:** twisting happens when the square sections of a branch rotate around the central
 443 axis. In some extreme cases of twisting, some sections of the branch may collapse to a single point.
- 444 • **Preserve section surface:** Along a branch, the surface of the section depends on the intermediate
 445 control points. We aim here at preserving the branch section surface, that is, the thickness of the
 446 branch, in order to keep the volume distribution along the branch as constant as possible.

447 Coherent orienting of a branch kite

448 We first define the geometry criteria we use to attach the kite K of a branch to the corresponding kite
 449 \bar{K} of the kite folio. In [Panotopoulou et al. \(2018\)](#), they propose to define a frame at each degree two vertex
 450 using Rotation Minimizing Frames (RMF); we use the same double-reflection algorithm proposed by [Wang](#)

451 [et al. \(2008\)](#) to determine the best matching between K and \overline{K} according to geometry. When a branch
 452 extremity K has been identified to \overline{K} and we traverse the branch for attaching the other side K' (see
 453 [Section 4.1](#)), we first compute the RMF frame from K to K' . We compute which points of K' and \overline{K}' best
 454 match by choosing the points correspondence minimizing the total distance between corresponding points.
 455 We consider identifying points of similar colors (that is, we choose one of the two possible configurations).
 456 When the branch is the first to be attached on the kite folio hosting K' , we also consider a red/blue swapping
 457 of the SQS (that leads to choosing the best configuration among four possible). This minimizes the twist
 458 at K' , thus along the branch. After this step, the parametric domain defined by the semi-simploidal set is
 459 entirely defined, and thus the set of control points to be embedded in 3D.

460 *Defining control points along a branch*

461 The control points of branch extremities are now positioned according to the kites, and the control points
 462 of the central Bézier curve c of the branch are mapped to the control point of the stick figure. In order to
 463 define the branch section around c , we need a frame along c to determine the set of control points to be
 464 embedded in 3D. We now determine the internal control points of the Bézier volume elements, aiming to
 465 ensure consistent volume distribution along the branch. A first idea is to place the internal control points
 466 of a branch by interpolating the control points between the starting and ending kites control points. This
 467 idea would be adapted for straight branches. For a non linear curve c , in order to prevent a flattening of the
 468 branch, the slices of control points need to be as orthogonal to the curve as possible – when a slice of control
 469 points becomes tangent to the curve, the corresponding part of the branch flattens. We thus define a local
 470 frame at each internal control point of c . Note that the rotation minimizing frames are a priori defined on
 471 the curve and not for the control points. Nevertheless the double-reflection algorithm [Wang et al. \(2008\)](#)
 472 only needs the tangent at a point to compute its local frame. In order to define a rotation minimizing frame
 473 for each internal control point we thus need a tangent at each control point. A Bézier control polygon with
 474 $n + 1$ control points converges smoothly to its curve [Morin and Goldman \(2001\)](#), but has n control tangents.
 475 We use degree elevation to associate a tangent to each control point: we can deduce $n + 1$ control tangents,
 476 thus, a frame per internal control points (see [Figure 7 \(a\) – \(c\)](#)). Interpolating the intermediate control
 477 points and expressing them into this local frame gives our branches a more intuitive (and not flattened)
 478 shape (see [Figure 7 \(d\)](#)).

479 This last step completes the automatic process for creating the volume model from the stick figure.

480 *4.3. Results*

481 [Figure 8](#) shows piecewise polynomial volumes we automatically generate from the stick figures on the
 482 left. We focus in particular in challenging situations regarding topology. The first example contains cycles,
 483 and all vertices of the stick graph are of degree at most 3. The second and third examples: a clover and an
 484 octopus has a central vertex of degree 9, and contains double edges. The last example, the snake, contains
 485 loops and curves of different degrees, and illustrates the smoothness of the proposed polynomial model.

486 **5. Conclusion and Future Work**

487 In this paper, we proposed a method for generating tubular piecewise polynomial volume objects from
 488 stick figures. We show that semi-simploidal sets can directly be used for handling both topology and
 489 geometry of volume tubular objects, providing a consistent structure together with a parametric domain for
 490 Bézier embedding.

491 We propose a branch structure made of four prisms which preserves the minimal number of cells on the
 492 object surface. If hexahedral volumes are needed, the presented approach still holds using a branch structure
 493 made of four hexahera (as cubes are also simploids). However, in this case, the boundary surface would
 494 have twice the number of quads.

495 The structure of junctions are defined by their corresponding SQS. In our approach, SQS are obtained by
 496 considering green points sequentially, so given a set of green points, different SQS may be obtained depending
 497 on the chosen order. It would be interesting to find a canonical SQS for a given set of green points, i.e.,

498 independent of ordering. In this case, a global optimization approach could be considered for defining the
 499 topology and the geometry of the sphere quadrangulation. We do not consider here the possible collisions
 500 between different parts of the model. It would be interesting to link geometric properties of the skeleton
 501 to prevent intersection of the model, and place the Bézier volume control points to avoid the geometric
 502 singularities despite the smoothness of the parameterized volume. We would also like to investigate varying
 503 thickness within the model. Branches could have different and varying thickness. This would implies an
 504 adaptation of junction handling.

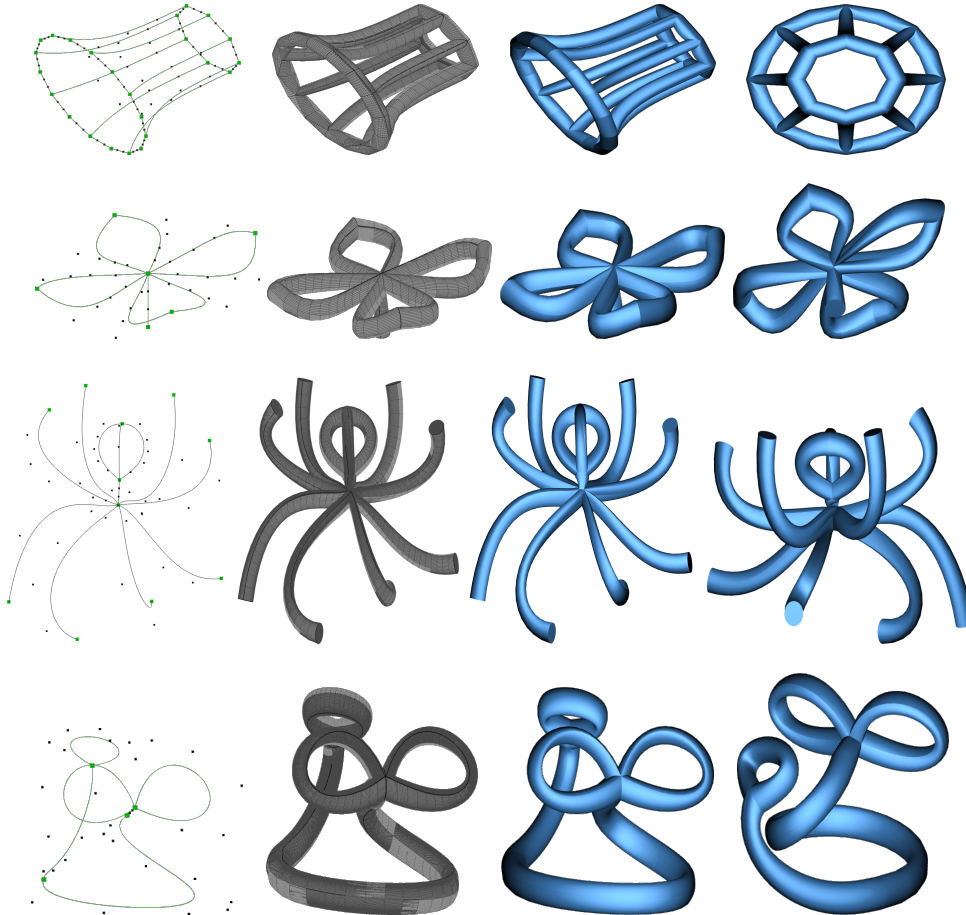


Figure 8: From left to right: stick figures made of Bézier curves in green; the control points are the black dots, then 3D volume meshes, volume object rendered, and another view of the same object. Even if performance is not a main keypoint of this work, we provide the computation time for this volume models: 750ms for the bin (first row), 117ms for the clover (second row), 80ms for the snake (third row) and 175ms for the octopus (last row).

505 **Acknowledgements**

506 The authors gratefully acknowledge Pascal Lienhardt for helpful discussions. This work benefited from
 507 government support managed by the National Research Agency under the *Investments for the future* program
 508 with the reference ANR-10-LABX-0074-01 Sigma-LIM.

509 **References**

510 Batagelj, V., 1989. An inductive definition of the class of 3-connected quadrangulations of the plane. *Discrete*
 511 *Mathematics* 78, 45–53. Special Double Issue in Memory of Tory Parsons.

- 512 Brinkmann, G., Greenberg, S., Greenhill, C., McKay, B.D., Thomas, R., Wollan, P., 2005. Generation of
513 simple quadrangulations of the sphere. *Discrete Mathematics* 305, 33–54. doi:[https://doi.org/10.](https://doi.org/10.1016/j.disc.2005.10.005)
514 [1016/j.disc.2005.10.005](https://doi.org/10.1016/j.disc.2005.10.005).
- 515 Brown, R., Higgins, P.J., 1981. On the algebra of cubes. *Journal of Pure and Applied Algebra* 21, 233 –
516 260.
- 517 Bærentzen, J., Misztal, M., Wehnicka, K., 2012. Converting skeletal structures to quad dominant meshes.
518 *Computers & Graphics* 36, 555 – 561. doi:<https://doi.org/10.1016/j.cag.2012.03.016>. shape
519 Modeling International (SMI) Conference 2012.
- 520 Cottrell, J.A., Hughes, T.J., Bazilevs, Y., 2009. *Isogeometric analysis: toward integration of CAD and FEA*.
521 John Wiley & Sons.
- 522 Dahmen, W., Micchelli, C.A., 1982. On the linear independence of multivariate b-splines I. Triangulation
523 of simploids. *SIAM J. Numer. Anal.* 19.
- 524 Damiand, G., Lienhardt, P., 2014. *Combinatorial Maps: Efficient Data Structures for Computer Graphics*
525 *and Image Processing*. A K Peters/CRC Press.
- 526 DeRose, T., Goldman, R.N., Hagen, H., Mann, S., 1993. Functional composition algorithms via blossoming.
527 *Transactions On Graphics* 12, 113–135.
- 528 Dobkin, D., Laszlo, M., 1987. Primitives for the manipulation of three-dimensional subdivisions, in: 3rd
529 Symposium on Computational Geometry, Waterloo, Canada. pp. 86–99.
- 530 Edmonds, J., 1960. A combinatorial representation for polyhedral surfaces. *Notices Amer. Math. Soc.* 7,
531 646.
- 532 Eilenberg, S., Zilber, J., 1950. Semi-simplicial complexes and singular homology. *Annals of Mathematics*
533 51, 499–513.
- 534 Farin, G., 2014. *Curves and surfaces for computer-aided geometric design: a practical guide*. Elsevier.
- 535 Fuentes Suárez, A., Hubert, E., 2018. Scaffolding skeletons using spherical voronoi diagrams: Feasibility,
536 regularity and symmetry. *Computer-Aided Design* 102, 83 – 93. doi:[https://doi.org/10.1016/j.](https://doi.org/10.1016/j.cad.2018.04.016)
537 [cad.2018.04.016](https://doi.org/10.1016/j.cad.2018.04.016). proceeding of SPM 2018 Symposium.
- 538 Guibas, L., Stolfi, G., 1985. Primitives for the manipulation of general subdivisions and the computation of
539 voronoi diagrams. *Transactions on Graphics* 4, 74–123.
- 540 Hatcher, A., 2002. *Algebraic Topology*. Cambridge University Press.
- 541 Hughes, T., Cottrell, J., Bazilevs, Y., 2005. Isogeometric analysis: Cad, finite elements, nurbs, exact
542 geometry and mesh refinement. *Computer Methods in Applied Mechanics and Engineering* 194, 4135–
543 4195. doi:<https://doi.org/10.1016/j.cma.2004.10.008>.
- 544 Lang, V., Lienhardt, P., 1996. Simplicial sets and triangular patches, in: *Computer Graphics International*
545 *Conference, CGI 1996, Pohang, Korea, June 24-28, 1996*, pp. 154–163.
- 546 Lienhardt, P., 1989. Subdivisions of n-dimensional spaces and n-dimensional generalized maps, in: *ACM*
547 *Symposium on Computational Geometry*, pp. 228–236.
- 548 Lienhardt, P., 1991. Topological models for boundary representation: a comparison with n-dimensional
549 generalized maps. *Computer-Aided Design* 23, 59–82. doi:[https://doi.org/10.1016/0010-4485\(91\)](https://doi.org/10.1016/0010-4485(91)90082-8)
550 [90082-8](https://doi.org/10.1016/0010-4485(91)90082-8).

- 551 Livesu, M., Ellero, S., Martínez, J., Lefebvre, S., Attene, M., 2017. From 3d models to 3d prints: an overview
552 of the processing pipeline, in: *Computer Graphics Forum*, Wiley Online Library. pp. 537–564.
- 553 Livesu, M., Muntoni, A., Puppo, E., Scateni, R., 2016. Skeleton-driven adaptive hexahedral meshing of
554 tubular shapes. *Computer Graphics Forum* 35, 237–246. doi:[10.1111/cgf.13021](https://doi.org/10.1111/cgf.13021).
- 555 Lyon, M., Bommers, D., Kobbelt, L., 2016. Hexex: robust hexahedral mesh extraction. *ACM Transactions*
556 *on Graphics (TOG)* 35, 1–11.
- 557 Mäntylä, M., 1988. *An Introduction to Solid Modeling*. Computer Science Press,.
- 558 Massarwi, F., Antolin, P., Elber, G., 2019. Volumetric untrimming: Precise decomposition of trimmed
559 trivariates into tensor products. *Computer Aided Geometric Design* 71, 1–15.
- 560 Massarwi, F., Elber, G., 2016. A b-spline based framework for volumetric object modeling. *Computer-Aided*
561 *Design* 78, 36–47.
- 562 Morin, G., Goldman, R., 2001. On the smooth convergence of subdivision and degree elevation for bézier
563 curves. *Computer Aided Geometric Design* 18, 657–666.
- 564 Munkres, J.R., 1984. *Elements of algebraic topology*. Perseus Books.
- 565 Nakamoto, A., 1996. Diagonal transformations in quadrangulations of surfaces. *Journal of Graph Theory*
566 21, 289–299. doi:[https://doi.org/10.1002/\(SICI\)1097-0118\(199603\)21:3<289::AID-JGT3>3.0.](https://doi.org/10.1002/(SICI)1097-0118(199603)21:3<289::AID-JGT3>3.0.CO;2-M)
567 [CO;2-M](https://doi.org/10.1002/(SICI)1097-0118(199603)21:3<289::AID-JGT3>3.0.CO;2-M).
- 568 Panotopoulou, A., Ross, E., Welker, K., Hubert, E., Morin, G., 2018. Scaffolding a skeleton, in: *Research*
569 *in Shape Analysis*. Springer, pp. 17–35.
- 570 Peltier, S., Fuchs, L., Lienhardt, P., 2009. Simplicoidal sets: Definitions, operations and comparison with
571 simplicial sets. *Discrete App. Math.* 157, 542–557 (extended version of "Homology of Simplicoidal Sets",
572 *DGCI 2006*, Szeged, Hungary, 235–246).
- 573 Peltier, S., Lienhardt, P., 2018. Simplicoidal Sets: a data structure for handling simplicoidal Bezier spaces.
574 *Computer Aided Geometric Design* 62, 44 – 62. doi:[10.1016/j.cagd.2018.03.010](https://doi.org/10.1016/j.cagd.2018.03.010).
- 575 Raptis, M., Kirovski, D., Hoppe, H., 2011. Real-time classification of dance gestures from skeleton animation,
576 in: *Proceedings of the 2011 ACM SIGGRAPH/Eurographics symposium on computer animation*, pp.
577 147–156.
- 578 Sanderson, C., Curtin, R., 2016. Armadillo: a template-based c++ library for linear algebra. *Journal of*
579 *Open Source Software* 1, 26.
- 580 Sundar, H., Silver, D., Gagvani, N., Dickinson, S., 2003. Skeleton based shape matching and retrieval, in:
581 *2003 Shape Modeling International*, IEEE. pp. 130–139.
- 582 Usai, F., Livesu, M., Puppo, E., Tarini, M., Scateni, R., 2016. Extraction of the quad layout of a triangle
583 mesh guided by its curve skeleton. *ACM Trans. Graph.* 35. doi:[10.1145/2809785](https://doi.org/10.1145/2809785).
- 584 Vince, A., 1983. Combinatorial maps. *Journal of Combinatorial Theory* 34, 1–21.
- 585 Wang, W., Jüttler, B., Zheng, D., Liu, Y., 2008. Computation of rotation minimizing frames. *ACM*
586 *Transactions on Graphics (TOG)* 27, 1–18.
- 587 Weiler, K., 1985. Edge-based data structures for solid modelling in curved-surface environments. *Comput.*
588 *Graph. Appl.* 5, 21–40.