Scaffolding skeletons using spherical Voronoi diagrams: feasibility, regularity and symmetry

Alvaro Javier Fuentes Suárez, Evelyne Hubert

Scaffolding skeletons using spherical Voronoi diagrams: feasibility, regularity and symmetry

A.J. Fuentes Suárez[1,2,*] and E. Hubert[1,2]

[1]INRIA Sophia Antipolis, France
[2]UCA, Sophia Antipolis, France

[*]Corresponding author: alvaro.fuentes-suarez@inria.fr

Abstract

Given a skeleton made of line segments we describe how to obtain a coarse quad mesh of a surface that encloses tightly the skeleton and follows its structure - the *scaffold*. We formalize as an Integer Linear Program the problem of constructing an optimal scaffold that minimizes the total number of quads on the mesh. We prove the feasibility of the Integer Linear Program for any skeleton. In particular we can generate these scaffolds for skeletons with cycles. We additionally show how to obtain *regular* scaffolds, i.e. with the same number of quad patches around each line segment, and *symmetric* scaffolds that respect the symmetries of the skeleton. An application to polygonization of skeleton-based implicit surfaces is also presented.

Keywords: Quad mesh generation, Skeleton based modeling, Procedural modeling, Scaffold.

1 Introduction

Skeletons are used in 3D graphics for modeling and animating articulated shapes. The user can design a complex shape by sketching a simple geometric object that is the input to a surface generating algorithm. By making changes in the skeleton it is possible to change the shape in an intuitive way. Due to their low dimensional nature, skeletons can serve as an efficient and compact representation of a surface. In this context a skeleton-based mesh generation method is needed.

The idea developed here is to construct a "coarse" quad mesh that tightly follows the structure of the skeleton. Following the terminology of [21] we call this process *scaffolding* and the corresponding coarse mesh a *scaffold*. This is used as an *intermediate* step in many applications. A scaffold can be used to generate a surface, either by subdivision as in [4], or as initial patchwork for a spline surface [7, 16] (see Figure 1). It is used as an intermediate step in the extraction of a quad layout on a given triangular mesh [5, 24], and for compatible quadrangulation [26]. An application we present here is the polygonization of skeleton-based implicit surfaces into quad-dominant meshes. In particular, we use our method for visualization of convolution surfaces [8, 27]. Our main contribution is in the theoretical foundations of an algorithm that computes a scaffold for any skeleton, independently of its topology, and with no user interaction.

In this paper we deal with skeletons made of line segments that do not intersect except at the endpoints, then called *joints*.

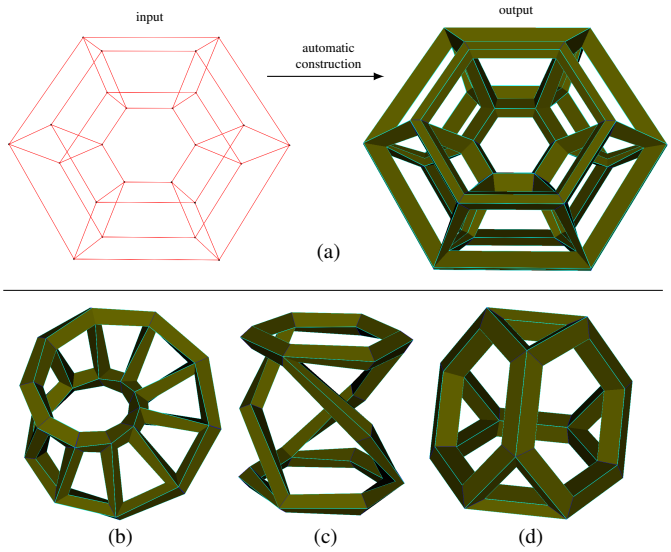


Figure 1: Recreation of some of the scaffolds used in [16], automatically computed by our method from suitable skeletons.

1.1 Previous work

One of the earliest ideas for a scaffold construction was to sweep a fixed polygonal cross profile along the segments, stitching the generated quads by means of a convex hull construction at the joints. This idea was first used in [23], producing meshes with some triangular faces resulting from the stitching process. For quadrilateral

cross profiles B-meshes method [15] improved upon this by merging triangles in order to get a quad-dominant mesh. Still the stitching process might leave some triangular patches at the joints.

[24] and [26] proposed a scaffolding technique that generates a pure-quad mesh by extruding boxes emanating from cubes positioned at the joints. In [24] the subdivision of the cubes is modeled with an Integer Linear Program, for which a solution might fail to exist when there are cycles in the skeleton. "Lids" [24, Figure 6] are introduced as a workaround. [26] also introduces extra quads around joints [26, Figure 4].

The method we propose is based on the Skeleton to Quad-dominant polygonal Mesh (SQM) method in [4] which was limited to skeletons without cycles. SQM first defines the mesh points around the joints and then recreate a quad based "tubular" polyhedral surface around each line segment. SQM can be regarded as a three steps process:

1. **Partition** the unit sphere centered at the joints into regions, one for each incident line segment.

2. **Discretize** each region into a cell (as an ordered set of points on its boundary) such that the two cells at the extremities of each line segment are *compatible*, i.e. have equal number of points. Points on the boundary of two regions are part of the two corresponding cells.

3. **Link**, in a bijective way, the cells at the extremities of each line segment. These links define the quads on the mesh.

For Step 3, SQM [4] defines the links by minimizing the total length of the line segments they define. In Step 2 Bærentzen *et al.* propose an algorithm for inserting additional vertices on the cells in such a way that the compatibility constraint is satisfied. Yet this algorithm does not work in the presence of cycles [4]. The existence of a possible discretization was actually not proved. Furthermore there was no analysis on the optimality of SQM with respect to the number of quads in the scaffold.

The partition of the sphere, in Step 1, used in [4] can be recognized to be a Voronoi diagram on the sphere. [21] introduces a partition of the sphere in quadrangles that makes the compatibility of cells (Step 2) trivial. Yet the partition is not canonical and the convexity of the regions is not guaranteed.

1.2 Contributions

Our method follows SQM [4] but we address and solve the crux difficulty that represents Step 2 for skeletons of arbitrary topology. We formalize the creation of compatible cells (Step 2) as an Integer Linear Program (IP) that minimizes the total number of quads. We prove that, for a Voronoi partition of the spheres, there exists a solution for the IP (feasibility), even in the presence of cycles. There is thus always an optimal solution that can be computed by an IP solver. Feasibility is proven thanks to a numerical characterization by Rivin [22] of graphs combinatorially equivalent to inscribable polyhedrons (i.e. with vertices on a sphere) that applies to the dual of the Voronoi diagram. Our minimization criterion is what ensures the coarsest mesh among those based on Voronoi partition of the spheres and additional pragmatic geometric restrictions. The solution of the IP determines the cross profile on each segment.

We present two other constructions to generate *symmetric* and *regular* scaffolds. In the former case the scaffold respects the symmetries of the skeleton. In the latter case, the scaffold has equal number of quads around each line segment of the skeleton, ensuring a similar cross profile for each line segment. Both possibilities are natural requirements for geometric modeling. With either or both requirements, we prove the feasibility of the constraints and are thus in a position to compute optimal solutions in the total number of quads.

To the extent of the knowledge of the authors the only paper that integrates the symmetries of the skeleton into the computation of the scaffold is [4], with the limitation that only one reflection symmetry can be taken into account for each skeleton (in addition to being restricted to cycle-free skeletons). Here we present a much more general approach that is able to compute scaffolds that respect any group of symmetries of the skeleton.

The article is organized as follows. In Section 2 we formalize the notions of skeleton and scaffold. In Section 3 we prove the existence of: standard, regular, and symmetric scaffolds, for any topology. In Section 4 we define an objective function and introduce the IPs that find optimal solutions. The algorithms to construct a scaffold are detailed in Section 5 with some further discussion in Section 6. An application to polygonization of skeleton-based implicit surfaces is shortly presented in Section 7.

A sketch of the first feasibility proof was presented at the conference LAGOS 2017. An extended abstract of the talk given there is available in the proceedings [13]. Here we generalize and extend the result in [13]. The precise objective function of the IP, the details of the algorithms, as well as the regular and symmetric cases are presented in this paper for the very first time.

2 Skeletons & Scaffolds

In this paper a *skeleton* is a finite set S of spatial line segments satisfying the following property: any two line segments intersect at most at one of their endpoints. A skeleton S defines naturally a graph $G_S = (\mathscr{V}_S, \mathscr{E}_S)$ by identifying the set of nodes $\mathscr{V}_S$ with the set of all endpoints in S, and the set of edges $\mathscr{E}_S$ with the set S itself. An edge $e \in \mathscr{E}_S$ connecting two nodes $a, b \in \mathscr{V}_S$ can be alternatively represented as ab. If $e = ab$ we say that e is *incident* to a (or b) and write $e \multimap a$ (or $e \multimap b$). A node with only one incident edge is called *dangling* node while a node with more than one incident edge is called a *joint*. A node with precisely two incident edges is called an *articulation*. We denote by $L_S \subset \mathscr{V}_S$ the set of all dangling nodes, $M_S \subset \mathscr{V}_S$ the set of all articulations, and $N_S = \mathscr{V}_S - (L_S \cup M_S)$ the set of the remaining joints.

The sphere centered at $v \in \mathscr{V}_S$ with radius $\varepsilon_v > 0$ is denoted $\mathbb{S}_v$, and $\mathscr{A}_v = \{e \cap \mathbb{S}_v \mid e \in \mathscr{E}_S, e \multimap v\}$ is the set of the points that are the intersection of the line segments incident to v with $\mathbb{S}_v$ (Figure 2b). For what follows the choices of ε_v ($v \in \mathscr{V}_S$) is independent of our method. We assume though that no two spheres intersect. Note that, once the scaffold mesh is created, $\max_{v \in \mathscr{V}_S} \varepsilon_v$ is an upper bound to the distance between any point in the edges of the scaffold and the skeleton.

For a joint $v \in \mathscr{V}_S$, the *Voronoi diagram* [3] of $\mathscr{A}_v$ on $\mathbb{S}_v$ (Figure 2c), denoted $\mathrm{Vor}(\mathscr{A}_v)$, partitions the sphere $\mathbb{S}_v$ into regions $\{R^v_e\}_{e \multimap v}$, with $(\mathbb{S}_v \cap e) \in R^v_e$, that are delimited by arcs of great circles [2, 20]. The cell C^v_e associated to the region R^v_e consists of the

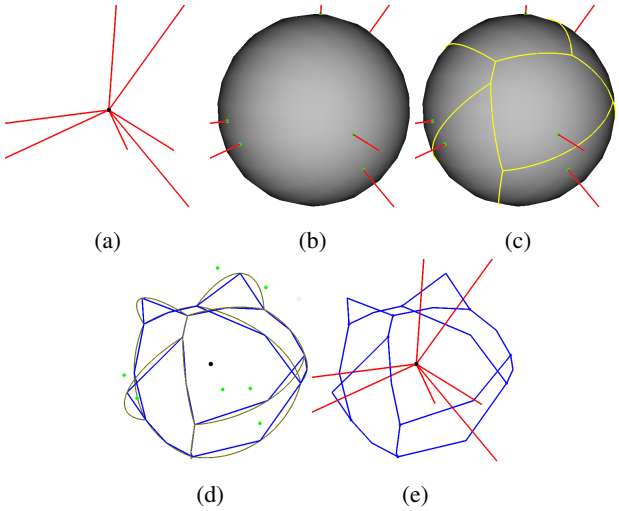


Figure 2: Construction of cells: for every joint (a) take the intersection of the incident segments with the unit sphere (b), compute the Voronoi diagram (c), subdivide the arcs in the boundaries of Voronoi regions (d) and take the ordered set of points (polyline) as representation of the cells (e).

end-points of the arcs delimiting R_e^v and some additional points (possibly none) per arc (Figure 2d). The points chosen in one arc define a polyline that represents the arc (Figure 2e). The number of segments in the polyline is called *number of subdivisions* of the arc, it is one less than the number of points taken in the arc.

One should also consider some geometrical constraints. The number of points in a cell must be at least 3 (4 is more customary [15, 21, 24, 26]). *Long arcs*, i.e. arcs with length greater than or close to π, must be subdivided into at least two segments. Examples of degenerate cases for cells with at least 3 or 4 points are shown in Figure 3.

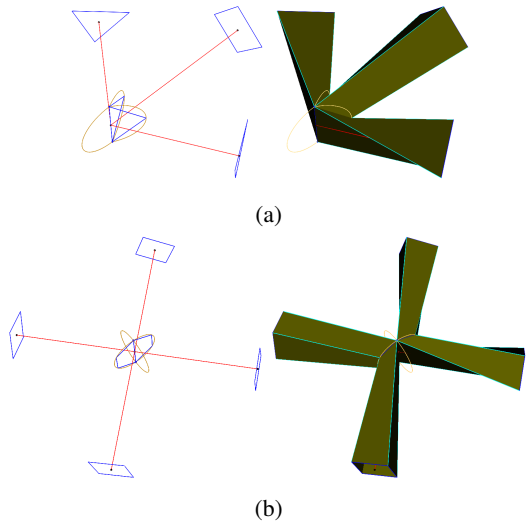


Figure 3: Degenerate cases arise when allowing for only one subdivision per arc, with at least 3 (a), or 4 (b), points on each cell. In both cases *long* pole-to-pole arcs are discretized as single segments going through the joint.

If the node v is an articulation, the arc separating the two Voronoi regions in $\mathscr{A}_v$ is a circle. Thus an associated cell C_e^v consists of points on the boundary circle. For v a dangling node, $\mathrm{Vor}(\mathscr{A}_v)$ consists of a single region with no boundary. In this case the associated cell C_e^v consists of points on the circle defined by the sphere $\mathbb{S}_v$ and the plane through v normal to e. In both cases the points on the cells are given by only one arc (great circle) and are taken such that the planar polygon they define encloses v.

There seems to be a preference in the literature for at least quadrangular cross profiles [15, 21, 24, 26]. To guarantee this the cells must have at least four points. We show in Figure 14 that three is also an adequate choice. In general, arcs can be discretized as a single segment (i.e. with no additional point) but the extra restrictions on the minimum number of points on each cell must be enforced to avoid cases like in Figure 3.

A *scaffold* $\mathscr{K}_S$ is defined as a pair (P_S, Φ_S), satisfying

1. $P_S = \{\mathscr{C}_v \mid v \in \mathscr{V}_S\}$, where each $\mathscr{C}_v = \{C_e^v \mid e \in \mathscr{E}_S, e \multimap v\}$ is a family of *cells* representing a partition of $\mathbb{S}_v$ according to $\mathrm{Vor}(\mathscr{A}_v)$.

2. $\Phi_S = \{\phi_e \mid e \in \mathscr{E}_S\}$ is a family of bijections ϕ_e between C_e^a and C_e^b for $e = ab$.

For $e = ab$ we say that C_e^a and C_e^b are *linked cells*. Similarly, if $C_e^a = \langle p_1, p_2, \ldots, p_n \rangle$ we say that p_i is *linked* with $\phi_e(p_i)$ and the pair $\langle p_i, \phi_e(p_i) \rangle$ is called a *link*. The realization of a scaffold as a mesh is through the quads defined by the four-points tuples $\langle p_i, \phi_e(p_i), \phi_e(p_{i+1}), p_{i+1} \rangle$.

To construct the links we follow the same strategy as in [4]: provided the cells have the same number of points, choose the bijection where the total length of the segments defined by the links is minimal.

The existence of the scaffold, for a given skeleton, is thus established if and only if we can discretize the Voronoi regions into compatible cells: we need the cells C_e^a and C_e^b (for all $e = ab \in \mathscr{E}_S$) to have the same number of points. This is far from obvious, specially in the presence of cycles in the skeleton.

3 Existence of scaffolds

We construct the set of linear equations over the integers that preside over the existence of a scaffold. The achievement is to prove the existence of a positive solution to this system, hence proving the existence of a scaffold for any given skeleton. The proof strongly relies on a property of Voronoi diagrams on the sphere, and more precisely on their duals. We examine this property first. We then prove the existence of a scaffold, which we qualify as *standard*. In geometric modeling it is desirable to have scaffolds that respect the symmetries of the underlying skeleton (*symmetric scaffold*), or to require a regularity on the number of quads around the line segments (*regular scaffold*). We can even seek scaffolds that satisfy both properties. We prove the existence of all these scaffolds.

3.1 Locally uniform discretization

The *Delaunay triangulation* $\mathrm{Del}(\mathscr{A}_v)$ is the dual of $\mathrm{Vor}(\mathscr{A}_v)$ [3]. For each $v \in \mathscr{V}_S$ let E_v be the set of edges of $\mathrm{Del}(\mathscr{A}_v)$. Each edge in E_v represents a common boundary between two regions in $\mathrm{Vor}(\mathscr{A}_v)$. For $f \in E_v$ we define a positive integer x_f^v representing the number

of subdivisions to be done to the corresponding arc (i.e. the number of segments in the polyline representation of the arc). For dangling nodes $E_v = \emptyset$, we nonetheless introduce a phantom edge ν_v, with an associated variable $x^v_{\nu_v}$, so that actually $E_v = \{\nu_v\}$.

The following lemma asserts that the Voronoi regions can be discretized uniformly, i.e. with an equal number of points. It is our main ingredient in proving the existence of scaffolds.

Lemma 3.1. *For $v \in \mathscr{V}_S$, the* local *linear system*

$$\sum_{\substack{f \in E_v \\ f \multimap (\mathbb{S}_v \cap e)}} x^v_f = \lambda_v \quad \forall e \multimap v, e \in \mathscr{E}_S \tag{1}$$

has a solution $(\tilde{x}^v_f, \tilde{\lambda}_v)$ with positive integer entries.

We denote the solution $(\tilde{x}^v_f, \tilde{\lambda}_v)$ as *local solution* associated to the *local* system (1).

To guarantee a positive solution we rely in a proposition due to Rivin [22] (statement extracted from [12]) that gives a numerical characterization for a graph of inscribable type (i.e. a graph combinatorially equivalent to a polyhedron inscribed on a sphere [12]).

Proposition 3.2 (I. Rivin)**.** *If a graph is of inscribable type then weights w can be assigned to its edges such that:*

(i) For each edge e, $0 < w(e) < 1/2$.

(ii) For each vertex v, the total weight of all edges incident to v is equal to 1.

Proposition 3.2 applies to $\mathrm{Del}(\mathscr{A}_v)$ that is combinatorially equivalent to the convex hull of $\mathscr{A}_v$ [10, 14], and hence of inscribable type. It thus guarantees a positive real solution for (1). Note that guaranteeing a positive solution for a linear system is not a trivial task.

The following claim then relates the existence of an integer positive solution of a homogeneous linear system to the existence of a real positive solution.

Proposition 3.3. *A homogeneous linear system with integer coefficients has a positive integer solution whenever it has a positive real solution.*

of Proposition 3.3. Let $Ay = 0$ be a homogeneous linear system where A is a $n \times m$ matrix with integer entries. Observe that if there is a rational solution p with $p \in \mathbb{Q}^m$, we can get an integer positive solution multiplying p by the least common multiple of denominators in the entries of p. Thus it is enough to prove that the system has a rational positive solution.

Let $\tilde{y} \in \mathbb{R}^m$ be the real solution of the homogeneous system with positive entries, this implies that the set of solutions of the system is a (non-trivial) subspace. Since A has integer entries we get a rational basis for the solution space. Let $\{y_1, y_2, \ldots, y_k\} \subset \mathbb{Q}^m$ $(k \geq 1)$ be a basis of the solution space of A. We have that $\tilde{y} = \sum_{i=1}^k \tilde{c}_i y_i$ for some real coefficients $\tilde{c}_i$. Let $f : \mathbb{R}^k \to \mathbb{R}^m$ be a function mapping $c \in \mathbb{R}^k$ to $f(c) = \sum_{i=1}^k y_i c_i$. Let $U = (0, \infty)^m$. Clearly U is open, and f is continuous, thus $V = f^{-1}(U) \subset \mathbb{R}^k$ is open. Moreover $\tilde{c} = (\tilde{c}_1, \ldots, \tilde{c}_k) \in V$ hence V is not empty. The set of rational points in $\mathbb{R}^k$ is dense, thus there exists $q \in \mathbb{Q}^k \cap V$. On the other hand $f(q) \in U$, that is all the entries of $f(q)$ are positive and rational and by definition $f(q)$ is in the solution space of A. Therefore $f(q)$ is a rational solution with positives entries for the system. □

of Lemma 3.1. For v a dangling node or articulation, a solution is trivially found. If v is a joint with at least three incident edges, the existence of a real positive solution for the local system (1) comes from the fact that $\mathrm{Del}(\mathscr{A}_v)$ is combinatorially equivalent to the convex hull of $\mathscr{A}_v$ [10, 14] which is an inscribed polyhedron. Thus Proposition 3.2 guarantees the existence of a real positive solution given by $x^v_f = w(f)$ and $\lambda_v = 1$, the result follows from Proposition 3.3. □

3.2 Standard scaffold

The number of points in a cell C^v_e ($e \in \mathscr{E}_S$ and $e \multimap v$) is given by

$$|C^v_e| = \sum_{\substack{f \in E_v \\ f \multimap (\mathbb{S}_v \cap e)}} x^v_f. \tag{2}$$

For each edge $e = ab \in \mathscr{E}_S$, there is a bijection $\phi_e \in \Phi_S$ in the scaffold $\mathscr{K}_S$ between the cells C^a_e and C^b_e. This is possible only if both cells have the same number of points, which gives the following compatibility equations

$$\sum_{\substack{h \in E_a \\ h \multimap (\mathbb{S}_a \cap e)}} x^a_h = \sum_{\substack{g \in E_b \\ g \multimap (\mathbb{S}_b \cap e)}} x^b_g \qquad \forall e = ab \in \mathscr{E}_S. \tag{3}$$

By definition x^v_f is a positive integer for all $v \in \mathscr{V}_S$, $f \in E_v$. In Section 2 we discussed some additional geometric constraints. They can be written in the form

$$\begin{cases} x^v_f \in \mathbb{Z}, x^v_f \geq 1 & \forall v \in \mathscr{V}_S, f \in E_v \\ \Lambda_i(x^v_f) \geq s_i & i = 1, 2, \ldots \end{cases} \tag{4}$$

where $\Lambda_i(x^v_f)$ are linear forms on the variables x^v_f with non-negative integer coefficients (not all zeros), and $s_i > 0$ are integer constants. These can capture the requirements of having at least 3 (or 4) points on each cell, as well as subdividing *long* arcs into at least two segments. The existence proof works for any set of (Λ_i, s_i) with non-negative coefficients. A practical realization of (4) for the geometric constraints commented in Section 2 and that should also serve as reference for the reader, is given by

$$\begin{cases} x^v_f \in \mathbb{Z}, x^v_f \geq m(x^v_f) & \forall v \in \mathscr{V}_S, f \in E_v \\ \sum\limits_{\substack{f \in E_v \\ f \multimap (\mathbb{S}_v \cap e)}} x^v_f \geq c & \forall v \in \mathscr{V}_S, e \in \mathscr{E}_S, e \multimap v \end{cases} \tag{5}$$

where c is 4 or 3; $m(x^v_f)$ is 1 if the length of the arc associated to x^v_f is less than $\pi - \delta$, and 2 otherwise. The choice of a small constant $\delta \in (0, \pi)$ determines the *long* arcs. Yet other constraints that may arise in applications, like subdividing specific arcs into a greater number of pieces, or requiring specific cells to have a greater number of points, can also be modeled in (4).

We say that the *global* system defined by (3) is *feasible* if it has a solution satisfying (4). Such a solution gives a way to discretize each region in the partition of the spheres into compatible cells, and thus allows to construct a scaffold. Proving the existence of a positive (integer or real) solution for a linear system is not a trivial task. The main result in our paper is the formal proof of feasibility of (3) which is stated in the following theorem.

Theorem 3.4. *For any skeleton S, the linear system given by* (3) *has a solution with the entries x_f^v ($v \in \mathscr{V}_S, f \in E_v$) satisfying the constraints given in* (4)*. Therefore there exists a scaffold for S.*

Proof. Using Lemma 3.1, for every $v \in \mathscr{V}_S$, we take $(\tilde{x}_f^v, \tilde{\lambda}_v)$ as a *local solution* satisfying (1). Then we take $\hat{x}_f^v = s\frac{\hat{\lambda}}{\tilde{\lambda}_v}\tilde{x}_f^v$ for all $v \in \mathscr{V}_S, f \in E_v$, where $\hat{\lambda} = \prod_{u\in\mathscr{V}_S}\tilde{\lambda}_u$ and $s = \max_i s_i$ is an integer constant. Using $\hat{x}_f^v$ as subdivisions for the arcs we get that all the cells have the same number of points: $s\hat{\lambda}$, it follows then that the equalities in (3) are trivially satisfied. The factors $s\hat{\lambda}/\tilde{\lambda}_v$ guarantee that the constraints in (4) are also satisfied. Thus $\hat{x}_f^v$ is a solution to (3) satisfying (4). □

The discretization constructed in the proof above is far from optimal. Optimality is dealt with in Section 4 with the help of Integer Linear Programming.

3.3 Symmetric scaffold

It is a desirable property for a scaffold to respect the symmetries of the underlying skeleton [4]. As illustrated in Figure 4 and 6, a standard scaffold need not satisfy this property. In this section we first define what is a valid symmetry for the skeleton. We then give the additional restrictions needed in order to obtain a scaffold that respects the symmetries of the skeleton.

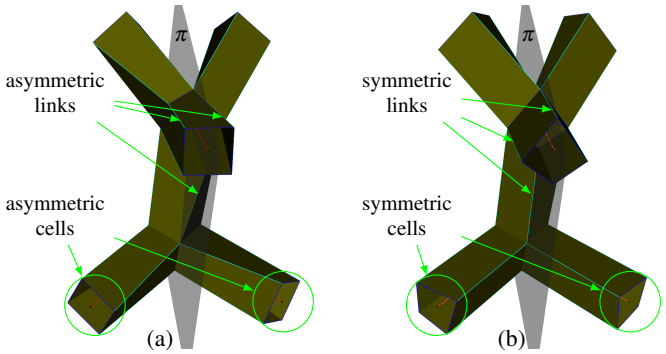


Figure 4: A symmetric scaffold is expected for a symmetric skeleton. In the picture the skeleton is symmetric through the plane π. (a) The cells of one standard scaffold does not respect the symmetry, links are not symmetric either. (b) The cells of a symmetric scaffold respect the symmetry.

A *skeleton symmetry* of S is an isometry $T : \mathbb{R}^3 \to \mathbb{R}^3$ such that

(i) $T(v) \in \mathscr{V}_S \quad \forall v \in \mathscr{V}_S$, and

(ii) $T(e) \in \mathscr{E}_S \quad \forall e \in \mathscr{E}_S$.

If the radii ε_v are different then in order to have a symmetric scaffold they must satisfy $\varepsilon_v = \varepsilon_{T(v)}$ for all $v \in \mathscr{V}_S, T \in \mathscr{T}_S$. We assume this is guaranteed for symmetric skeletons.

Conditions ((i)) and ((ii)) say that T keeps the set of nodes $\mathscr{V}_S$ and edges $\mathscr{E}_S$ (hence G_S) invariant, thus the whole skeleton is kept fixed: $T(S) = S$. Since T is an isometry and $e = ab \in \mathscr{E}_S$ is the line segment between the nodes a and b (including both), then $T(e)$ is the edge (line segment) connecting $T(a)$ and $T(b)$.

With conditions ((i)) and ((ii)) we avoid cases, as illustrated in Figure 5, where there is a geometric symmetry for S that does not map elements of G_S into elements of G_S. Thus not all symmetries are *skeleton symmetries*.

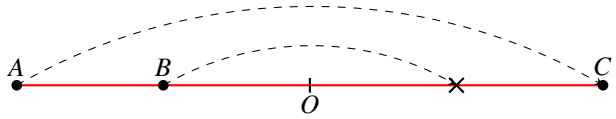


Figure 5: A skeleton with a false symmetry: the central symmetry with respect to O, maps the skeleton to itself when considered as a curve but not as a graph (B is not mapped to another node).

The set of skeleton symmetries $\mathscr{T}_S$ forms a finite group under composition. For simplicity we denote $T \circ R$ as TR.

We say then that a scaffold $\mathscr{K}_S = (P_S, \Phi_S)$ respects the skeleton symmetry $T \in \mathscr{T}_S$ if

$$C_{T(e)}^{T(v)} = T(C_e^v) \quad \forall v \in \mathscr{V}_S, e \in \mathscr{E}_S, e \multimap v, \tag{6}$$

and

$$\phi_{T(e)} = T \circ \phi_e \circ T^{-1} \quad \forall e \in \mathscr{E}_S. \tag{7}$$

Since Voronoi diagrams depend only on the distance between points, it follows that

$$\mathrm{Vor}(\mathscr{A}_{T(v)}) = T(\mathrm{Vor}(\mathscr{A}_v)). \tag{8}$$

Notice that it is also true that $\mathscr{A}_{T(v)} = T(\mathscr{A}_v)$ and $E_{T(v)} = T(E_v)$. Hence Equation (6) can be achieved as soon as the subdivisions are done in a symmetric way for symmetric arcs. This is possible if

$$x_f^v = x_{T(f)}^{T(v)} \qquad \forall T \in \mathscr{T}_S, v \in \mathscr{V}_S, f \in E_v. \tag{9}$$

To guarantee (6) given (9), it is sufficient to subdivide the arcs into equal length subdivisions since arc-length is preserved under isometries: the arc corresponding to $f \in E_v$ ($v \in \mathscr{V}_S$) with arc-length θ is subdivided into x_f^v sub-arcs of length θ/x_f^v. Equation (7) means that the links defined by the bijections in Φ_S define symmetric line segments, and hence symmetric quads.

Solutions of (3) satisfying (4) and the extra constraints given by (9), yield compatible cells that respect the skeleton symmetries. The existence of such a solution is denoted as feasibility of the symmetric scaffold, and it is established in Theorem 3.6.

The proof of Theorem 3.6 relies on the following lemma.

Lemma 3.5. *Let $\hat{x}_f^v$ be a solution of the linear system given by* (3) *satisfying the constraints in* (4)*. Then $\bar{x}_f^v = \sum_{T\in\mathscr{T}_S} \hat{x}_{T(f)}^{T(v)}$ is also a solution of* (3) *satisfying* (9) *and* (4)*.*

Proof. Let $T \in \mathscr{T}_S$. Since $\mathscr{T}_S$ is a group we have $\mathscr{T}_S = \{RT \mid R \in \mathscr{T}_S\}$. Hence

$$\bar{x}_{T(f)}^{T(v)} = \sum_{R\in\mathscr{T}_S} \hat{x}_{RT(f)}^{RT(v)} = \sum_{R\in\mathscr{T}_S} \hat{x}_{R(f)}^{R(v)} = \bar{x}_f^v. \tag{10}$$

Thus $\bar{x}_f^v$ satisfies (9).

We prove now that $\bar{x}_f^v$ is a solution of the linear system given by (3). For $e = ab \in \mathscr{E}_S$ we have $T(E_a) = E_{T(a)}$ because of the

symmetric property of Voronoi diagrams (8), and hence of its dual. Therefore

$$\sum_{\substack{h\in E_a \\ h\multimap(\mathbb{S}_a\cap e)}} \hat{x}^{T(a)}_{T(h)} = \sum_{\substack{k\in E_{T(a)} \\ k\multimap(\mathbb{S}_{T(a)}\cap T(e))}} \hat{x}^{T(a)}_{k}. \tag{11}$$

Similarly

$$\sum_{\substack{g\in E_b \\ g\multimap(\mathbb{S}_b\cap e)}} \hat{x}^{T(b)}_{T(g)} = \sum_{\substack{l\in E_{T(b)} \\ l\multimap(\mathbb{S}_{T(b)}\cap T(e))}} \hat{x}^{T(b)}_{l}. \tag{12}$$

T is a skeleton symmetry, thus $T(e) \in \mathscr{E}_S$ is the edge connecting $T(a)$ and $T(b)$. Since $\hat{x}^v_f$ is a solution of the system given by (3), taking the equation for the edge $T(e)$ in (3), it follows that

$$\sum_{\substack{k\in E_{T(a)} \\ k\multimap(\mathbb{S}_{T(a)}\cap T(e))}} \hat{x}^{T(a)}_{k} = \sum_{\substack{l\in E_{T(b)} \\ l\multimap(\mathbb{S}_{T(b)}\cap T(e))}} \hat{x}^{T(b)}_{l}. \tag{13}$$

From (11), (12) and (13) we get

$$\sum_{\substack{h\in E_a \\ h\multimap(\mathbb{S}_a\cap e)}} \hat{x}^{T(a)}_{T(h)} = \sum_{\substack{g\in E_b \\ g\multimap(\mathbb{S}_b\cap e)}} \hat{x}^{T(b)}_{T(g)} \quad \forall e \in \mathscr{E}_S. \tag{14}$$

Summing the latter equation over $T \in \mathscr{T}_S$ proves that $\bar{x}^v_f$ is a solution of (3). □

Theorem 3.6. *For any skeleton S with a group of symmetries $\mathscr{T}_S$, the linear system given by* (3) *has a solution with the entries x^v_f $(v \in \mathscr{V}_S, f \in E_v)$ satisfying the constraints given in* (4) *and* (9)*. Therefore there exists a symmetric scaffold for S.*

Proof. By Theorem 3.4 we have that there is a solution $\hat{x}^v_f$ of (3) satisfying (4). Lemma 3.5 then gives a solution $\bar{x}^v_f$ to (3) satisfying (9). Since $\bar{x}^v_f \geq \hat{x}^v_f$, we have that (4) is trivially satisfied by $\bar{x}^v_f$. □

An example of a skeleton with a rotation symmetry is shown in Figure 6. If one is not interested in a solution that respects all the symmetries of the skeleton, one can restrict the set $\mathscr{T}_S$ to be the group generated by a subset of the skeleton symmetries.

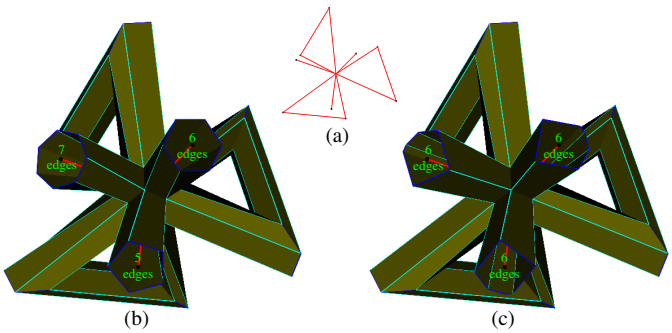


Figure 6: A three-fold rotation symmetry (C_3). (a) Skeleton. (b) Standard scaffold. (c) Symmetric scaffold.

3.4 Regular symmetric scaffold

We call a scaffold *regular* if it has the same number of quads around each line segment. This is not an automatic property of *standard* scaffolds, as can be seen in figure 6b and 7.

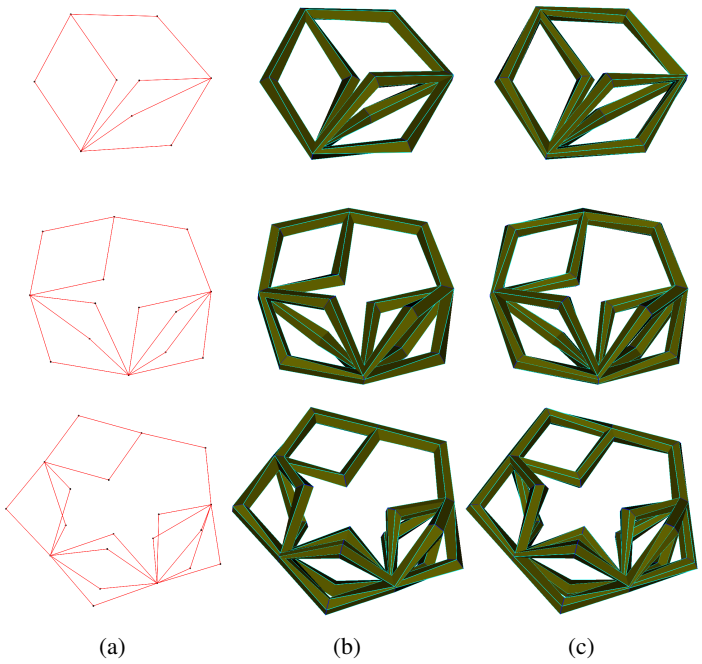


Figure 7: Complex closed skeletons (a), standard scaffolds (b), and regular scaffolds (b)

To get regular scaffolds we need compatible cells that have the same number of points, this means that (3) must be replaced by

$$\sum_{\substack{f\in E_v \\ f\multimap(\mathbb{S}_v\cap e)}} x^v_f = \lambda \quad \forall v \in \mathscr{V}_S, e \multimap v, e \in \mathscr{E}_S \tag{15}$$

with λ an additional free variable. Notice that λ is independent of $v \in \mathscr{V}_S$ and so (15) implies (3). Solutions to the linear system (15) satisfying (4) are called *regular* solutions. Regularity ensures that all segments have a similar cross profile: a polygon with λ sides. If there is a regular solution we say that (15) is feasible, which implies the existence of a regular scaffold.

It is possible to get a scaffold that is at the same time regular and symmetric. We just need to ensure, besides (4), the extra constraints in (9). A regular scaffold need not be symmetric (Figure 8a). Conversely, a symmetric scaffold is not necessarily regular (Figure 8b).

The existence of regular symmetric scaffolds is established in the following theorem. It is sufficient to prove the feasibility of *regular symmetric* scaffolds to automatically get the feasibility of *regular* scaffolds. Indeed a regular scaffold can be regarded as a regular symmetric scaffold with $\mathscr{T}_S$ the trivial group consisting of only the identity symmetry.

Theorem 3.7. *For any skeleton S admitting a group of symmetries $\mathscr{T}_S$, the linear system given by* (15) *has a solution with the entries x^v_f $(v \in \mathscr{V}_S, f \in E_v)$ satisfying the constraints given in* (4) *and* (9)*. Therefore there exists a regular symmetric scaffold for S.*

Proof. As done in the proof of Theorem 3.4, for every $v \in \mathscr{V}_S$ Lemma 3.1 gives a *local solution* $(\tilde{x}^v_f, \tilde{\lambda}_v)$ satisfying (1). Then $\hat{x}^v_f = s\frac{\hat{\lambda}}{\tilde{\lambda}_v}\tilde{x}^v_f$ with $\hat{\lambda} = \prod_{u\in\gamma_S}\tilde{\lambda}_u$ and $s = \max_i s_i$, is a local solution such that all the cells have $s\hat{\lambda}$ points, thus $\hat{x}^v_f$ is a global solution of (15) satisfying (4). Applying Lemma 3.5 to this solution we get a

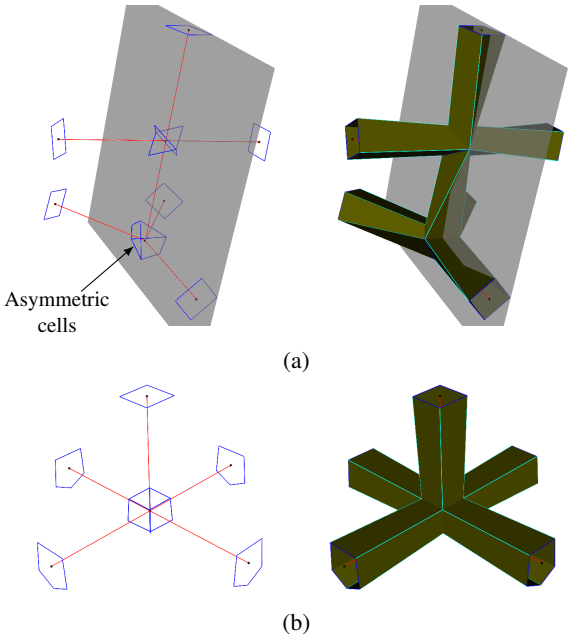


Figure 8: Symmetry and regularity are independent properties. (a) Regular asymmetric scaffold: the highlighted cell is asymmetric with respect to the reflection symmetry. (b) Symmetric irregular scaffold: one edge has a quadrangular cross profile while the rest are pentagonal.

symmetric solution $\bar{x}_f^v$ with the same number of points on each cell. Indeed

$$|\bar{C}_e^v| = \sum_{\substack{f\in E_v \\ f\multimap(\mathbb{S}_v\cap e)}} \bar{x}_f^v = \sum_{\substack{f\in E_v \\ f\multimap(\mathbb{S}_v\cap e)}} \sum_{T\in\mathscr{T}_S} \hat{x}_{T(f)}^{T(v)} =$$

$$= \sum_{T\in\mathscr{T}_S} \sum_{\substack{g\in E_{T(v)} \\ g\multimap(\mathbb{S}_{T(v)}\cap T(e))}} \hat{x}_g^{T(v)} = \sum_{T\in\mathscr{T}_S} |\hat{C}_{T(e)}^{T(v)}| = s\hat{\lambda}\,|\mathscr{T}_S|.$$

□

Symmetry is an important property for the scaffolds as can be appreciated in Figure 9 computed for one of the examples in [16].

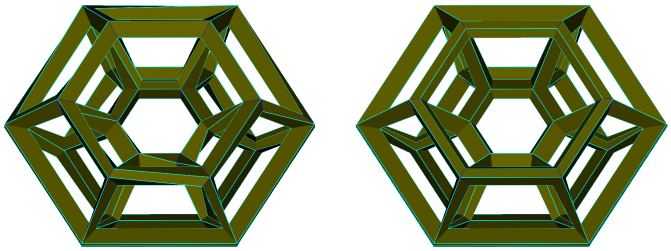


Figure 9: With symmetric requirements a scaffold can be greatly improved. (a) The computation of the scaffold in Figure 1a without symmetric requirements. (b) The "correct" symmetric scaffold respecting all the symmetries of the skeleton (same as Figure 1a).

4 Optimal scaffolds

We have established that for any skeleton S admitting a group of symmetries $\mathscr{T}_S$, there exist a standard scaffold, a symmetric scaffold, a regular scaffold, and a regular symmetric scaffold. In this section we address the computation of optimal scaffolds. We model each scaffolding problem as an Integer Linear Program (IP) [11] that looks for a solution with a minimal number of quads. For this we need first to define an objective function, then the constraints of the IPs. The existence of a minimal solution is guaranteed by the existence theorems on Section 3, thus such minimal solutions can be computed with an IP solver.

4.1 Objective function

We want to express the total number of quads in a scaffold in terms of the subdivision of the arcs. For this we proceed as follows. Let $\mathscr{Q}$ be the set of all quads of a scaffold, and P the number of pairs $\langle p, Q\rangle$ such that $p \in Q \in \mathscr{Q}$. Then we have that

$$|\mathscr{Q}| = \tfrac{1}{4}P. \tag{16}$$

Indeed, the last equation is a consequence of the following observation: every fixed quad Q has 4 points and there are precisely 4 pairs $\langle q, Q\rangle$ with $p \in Q$.

We can count P in another way: by fixing first a point and then counting the pairs containing it. For a point p in the cell of a dangling node, there are 2 quads containing p. If p is a point in the cell of an articulation, there are 4 quads containing p. If p is a point in an arc on the boundary of a Voronoi region of a joint (not an articulation), we get that p is in 4 quads if p is not an extremity of the arc, or p is in $2d(p)$ quads if it is an extremity of an arc. Here $d(p)$ denotes the number of Voronoi regions that have p in their common boundary.

The latter paragraph can be summarized as

$$P = \sum_{\substack{v\in L_S \\ f\in E_v}} 2x_f^v + \sum_{\substack{v\in M_S \\ f\in E_v}} 4x_f^v + \sum_{\substack{v\in L_S \\ f\in E_v}} 4(x_f^v - 1) + \Theta_S, \tag{17}$$

where Θ_S is the number of pairs containing points that are extremities of the arcs in the Voronoi diagrams. Notice that Θ_S is a constant for a fixed skeleton S, thus (17) can be written as

$$P = \sum_{\substack{v\in L_S \\ f\in E_v}} 2x_f^v + \sum_{\substack{v\in M_S \\ f\in E_v}} 4x_f^v + \sum_{\substack{v\in L_S \\ f\in E_v}} 4x_f^v + \Phi_S, \tag{18}$$

where $\Phi_S = \Theta_S - \sum_{v\in L_S} 4|E_v|$ is a constant for a fixed skeleton S.

From equations (16) and (18) we conclude that

$$\sum_{v\in(\gamma_S - L_S)} \sum_{f\in E_v} 2x_f^v + \sum_{v\in L_S} \sum_{f\in E_v} x_f^v \tag{19}$$

is an objective function that minimizes the total number of quads in a scaffold.

4.2 Integer Linear Programming models

We can now state the Integer Linear Programs (IPs) that compute subdivision for the arcs such that the scaffold have a minimal number of quads. For all the IPs the optimality criterion is to minimize the objective function (19). In Table 1 we show each model.

Scaffold	Constraints	Minimize function
Standard	(3), (4)	(19)
Regular	(15), (4)	(19)
Symmetric	(3), (4), (9)	(19)
Regular Symmetric	(15), (4), (9)	(19)

Table 1: Integer Linear Program models.

Theorems 3.4, 3.6 and 3.7 prove the feasibility of the standard, symmetric, and regular symmetric models respectively. As discussed in Section 3.4, the feasibility of the regular IP model is a consequence of the feasibility of the regular symmetric model.

A final observation is that once we can guarantee feasibility, and since all the variables in (19) are positive integers, then an optimal minimal solution always exists. The optimal solution can be computed with a Mixed-Integer Linear Program Solver. In particular with the branch-and-cut [11, Chapter 12] implementation of GLPK [19].

5 Algorithms

In this section we provide practical algorithms for the implementation of our method. Although our main contribution is in the theoretical foundation and the proofs of the existence of scaffolds, the algorithms in this section show that our method can be readily implemented to compute scaffolds for any skeleton.

The general steps for an implementation are described in Algorithm 1. Sub-algorithms 2, 3 and 4 deal with the details of the construction of compatible cells from the output of the standard IP and the definition of the bijections between cells (linking process).

Input: The set of nodes $\mathscr{V}_S$ and edges $\mathscr{E}_S$ representing the skeleton.
Output: The quads that represent a scaffold.

1 **foreach** *node* $v \in \mathscr{V}_S$ **do**
2 $\quad \mathscr{A}_v \leftarrow \{e \cap \mathbb{S}_v : e \in \mathscr{E}_S, e \multimap v\}$;
3 $\quad \mathscr{H}_v \leftarrow convexHull(\mathscr{A}_v)$;
4 $\quad E_v \leftarrow edgesOf(\mathscr{H}_v)$;
5 Define and solve the Linear Program for the subdivision of arcs; `/* Alg. 2 */`
6 Construct compatible cells; `/* Alg. 3 */`
7 Define bijections of linked cells; `/* Alg. 4 */`
8 **foreach** *edge* $e = ab \in \mathscr{E}_S$ **do**
9 $\quad$ Let $C_e^a = \langle p_0, p_2, \ldots, p_n \rangle$;
10 $\quad$ **for** $i = 0$ **to** n **do**
11 $\quad\quad$ Output quad $\langle p_i, \phi_e(p_i), \phi_e(p_{i+1}), p_{i+1} \rangle$ `/* i+1 is taken mod n */`

Algorithm 1: General algorithm for constructing a scaffold.

Algorithm 1 follows the general three steps we introduced in Section 1. In Step 1 we use spherical Voronoi diagrams for the partition of the sphere at the joints (lines 1–4). We mostly work with their duals, Delaunay triangulations, which are equivalent to the the convex hulls [10, 14]. The discretization of the regions, Step 2, is divided into sub-algorithms.

The convex hull computed in Algorithm 1 (line 3) is not always 3-dimensional. It may be a 2-dimensional convex hull (a polygon), a 1-dimensional convex hull (one edge), or a 0-dimensional one (a point). Those cases correspond respectively to: more than three points in general position, more than two coplanar points, two points, and only one point (Figure 10). 1-dimensional convex hulls occur around articulations, while 0-dimensional ones are associated to dangling nodes. In our implementation the convex hull is computed by means of the QHull library [6].

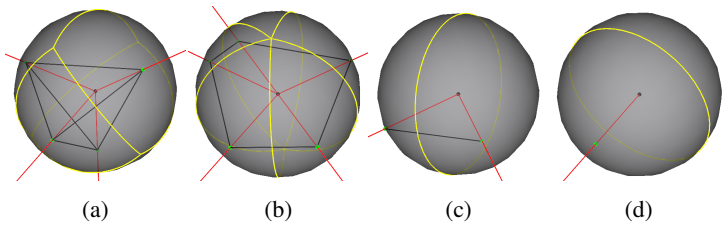


Figure 10: Cases for the convex hull of $\mathscr{A}_v$. Edges of E_v are shown in black. (a) 3-dimensional: more than three points in general position. (b) 2-dimensional: more than two coplanar points. (c) 1-dimensional: two points. (d) 0-dimensional: one point.

Algorithm 2 deals with the standard IP whose optimal solutions give compatible cells. It sets up and solves the IP using the convex hull representation of the regions. Simple modifications can be done for the other three variants of the scaffold. We follow (5). Thus we chose to have at least 4 points on each cell to guarantee at least quadrangular cross sections as in [4, 15, 21, 24, 26]. We set the threshold for *long* arcs at $\frac{5\pi}{6}$ (i.e $\delta = \frac{\pi}{6}$), above which at least two subdivisions must be done. In our implementation we solve the IP with the branch-and-cut [11, Chapter 12] implementation in GLPK [19].

Input: The set of nodes $\mathscr{V}_S$ and edges $\mathscr{E}_S$ representing the skeleton, along with E_v for each $v \in \mathscr{V}_S$.
Output: The values x_f^v representing the number of subdivisions for each arc that gives compatible cells.

1 Initialize the linear program IP;
2 **foreach** *node* $v \in \mathscr{V}_S$, *and edge* $f \in E_v$ **do**
3 $\quad$ Add integer variable x_f^v to IP;
4 $\quad$ **if** *the arc associated to* x_f^v *has length* $< \frac{5\pi}{6}\varepsilon_v$ **then**
5 $\quad\quad$ Add restriction $x_f^v \geq 1$ to IP;
6 $\quad$ **else**
7 $\quad\quad$ Add restriction $x_f^v \geq 2$ to IP;
8 **foreach** *pair* (v,e) *with* $v \in \mathscr{V}_S$, $e \in \mathscr{E}_S$ *and* $e \multimap v$ **do**
9 $\quad$ Add following restriction to IP $\sum_{\substack{f \in E_v \\ f \multimap (e \cap \mathbb{S}_v)}} x_f^v \geq 4$;
10 **foreach** *edge* $e = ab \in \mathscr{E}_S$ **do**
11 $\quad$ Add following restriction to IP $\sum_{\substack{g \in E_a \\ g \multimap (e \cap \mathbb{S}_v)}} x_g^a = \sum_{\substack{h \in E_b \\ h \multimap (e \cap \mathbb{S}_v)}} x_h^b$;
12 Define objective function $\sum_{v \in (\mathscr{V}_S - L_S)} \sum_{f \in E_v} 2x_f^v + \sum_{v \in L_S} \sum_{f \in E_v} x_f^v$ for IP;
13 Solve IP by minimizing the objective function.

Algorithm 2: Compute subdivisions for compatible cells.

Algorithm 3 computes the cells from the subdivision numbers and the convex hull representation of the Voronoi diagram, and such it must handle the different cases of the convex hulls (see Figure 10).

The heuristic for the linking process (Step 3) is described in Algorithm 4, from which the resulting quads are defined. It defines the

Input: Nodes $\mathscr{V}_S$ and edges $\mathscr{E}_S$ of the skeleton, E_v, $\mathscr{H}_v$ and the subdivision numbers x_f^v.
Output: A list $\mathscr{C}$ of compatible cells for the scaffold.

$\mathscr{C} \leftarrow emptyList()$;
foreach $v \in \mathscr{V}_S$ **do**
  if $\mathscr{H}_v$ *is 3-dimensional* **then**
    /* at least 4 not coplanar nodes */
    foreach *node* n_e *in* $\mathscr{H}_v$ **do** /* $n_e = e \cap \mathbb{S}_v$ for $e \multimap v$ */
      $C_e^v \leftarrow emptyList()$;
      foreach $f \multimap (e \cap \mathbb{S}_v)$ **do**
        Let F_1, F_2 be the two faces in $\mathscr{H}_v$ that have common boundary f;
        Compute the outward pointing unit normals N_1, N_2 of F_1, F_2 respectively;
        Compute $x_f^v + 1$ points in the arc from $v + N_1$ to $v + N_2$ going perpendicularly to f on $\mathbb{S}_v$;
        Add the points to C_e^v avoiding repetitions;
      Add C_e^v to $\mathscr{C}$;
  if $\mathscr{H}_v$ *is 2-dimensional* **then**
    /* at least 3 nodes, all coplanar */
    Let N_1, N_2 be the two unit normals of the plane supporting $\mathscr{H}_v$;
    foreach *node* n_e *in* $\mathscr{H}_v$ **do** /* $n_e = e \cap \mathbb{S}_v$ for $e \multimap v$ */
      $C_e^v \leftarrow emptyList()$;
      Let f, g be the two edges incidents to n_e;
      Compute $x_f^v + 1$ points in the arc from $v + N_1$ to $v + N_2$ going perpendicularly to f on $\mathbb{S}_v$;
      Compute $x_g^v + 1$ points in the arc from $v + N_1$ to $v + N_2$ going perpendicularly to g on $\mathbb{S}_v$;
      Add the points to C_e^v avoiding repetitions;
      Add C_e^v to $\mathscr{C}$;
  if $\mathscr{H}_v$ *is 1-dimensional* **then**
    /* only one edge and 2 nodes */
    Let f be the unique edge in E_v;
    Compute x_f^v points in the circle with center v and perpendicular to f on $\mathbb{S}_v$;
    Add the points to C_e^v;
    Add C_e^v to $\mathscr{C}$;
  if $\mathscr{H}_v$ *is 0-dimensional* **then**
    /* only one node */
    Compute $x_{\mathscr{V}_v}^v$ points in the great circle with center v and perpendicular to e on $\mathbb{S}_v$;
    Add the points to C_e^v;
    Add C_e^v to $\mathscr{C}$;
return $\mathscr{C}$

Algorithm 3: Construct compatible cells.

bijections (links) by minimizing the total length of the links (same heuristic used in [4]).

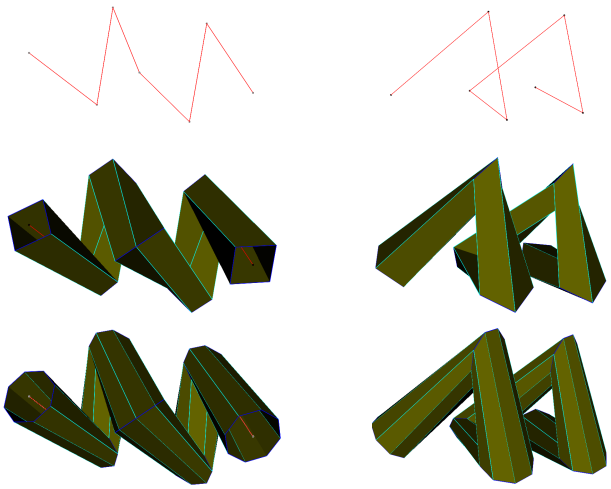
Input: The set of compatible cells $\mathscr{C} = \{C_e^v \mid v \in \mathscr{V}_S, e \in \mathscr{E}_S\}$.
Output: The bijections ϕ_e between linked cells.

foreach *edge* $e = ab \in \mathscr{E}_S$ **do**
  Order points in the cells C_e^a and C_e^b according to the angle around the edge e;
  Let $C_e^a = \langle p_0, p_2, \ldots, p_n \rangle$ and $C_e^a = \langle q_0, q_2, \ldots, q_n \rangle$;
  $D_{min} \leftarrow \sum_{i=0}^{n} \|p_i - q_i\|$; $k \leftarrow 0$;
  for $j = 1$ **to** $n-1$ **do**
    $D \leftarrow \sum_{i=0}^{n} \|p_i - q_{i+j}\|$;
    if $D < D_{min}$ **then**
      $D_{min} \leftarrow D$; $k \leftarrow j$;
  Define bijection the ϕ_e as $p_i \mapsto q_{i+k}$;
  /* $i+k$ is taken mod n */

Algorithm 4: Construct bijections between linked cells.

To better reflect the geometry of the model, and depending on the application at hand, the position of points in the discretization of each arc can be modified by a Laplacian smoothing [9] or another heuristic. A global optimization could be further applied to take into account the twisting of the quads around a chain of articulation nodes (Figure 11). The challenge is in devising such a "better" positioning of points that also respects the symmetries of the skeleton. An intrinsic solution using our method is to subdivide each arc twice (or more) as shown in Figure 11.

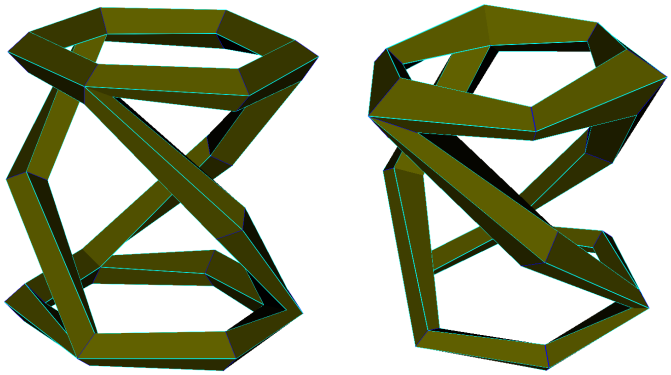
An example of scaffold with different radii for the spheres at the joints is shown in Figure 12. Note that the skeleton is symmetric as well as the scaffold, hence the values of the radii are also symmetric.



(a) (b)

Figure 11: In a chain of articulations the scaffold might have a twisting behavior: (a) and (b) – different views of the same model; top row – the skeleton; middle row – the standard scaffold; and bottom row – the scaffold with extra subdivisions. Note how in the bottom row the twisting behavior of the quads is diminished.

Symmetric and regular scaffolds Regular scaffolds can be obtained with the algorithms described above, a simple variation of



(a) (b)

Figure 12: A scaffold with different radii for the spheres at the joints: (a) the original scaffold, (b) the scaffold with varying radii.

Algorithm 2 is needed in order to account for the restrictions given in (15). Similarly, for symmetric scaffolds we need to add the extra restrictions in (9), taking special care with the definition of links (Algorithm 4). Dangling nodes or articulations fixed by the skeleton symmetries might present issues if the linking is naively done. In our implementation we modify Algorithm 4 such that the links are propagated from one edge to its symmetric ones. The cells of dangling nodes are computed by projecting the linked cells onto the plane perpendicular to the incident edge (of the dangling nodes). This guarantees that the links respect the symmetries.

Linear system simplification The IP in Algorithm 2 can be simplified by removing variables and equations relative to dangling nodes. The equations in a chain of articulations can be merged into one equation relating the extremal cells of the chain, removing the variables relative to intermediate articulations. In Figure 13 we illustrate the simplification cases.

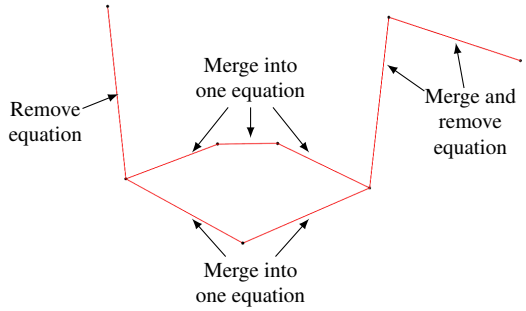


Figure 13: Simplification of the equations in the IP.

The general complexity of our method is bounded by the IP solver algorithm, which theoretically is NP-complete [17]. In practice, as usual with Linear Programming, the solver behaves well. The convex hull implementation has average complexity $n \log n$, with n the number of skeleton edges. In Table 2 we show a summary of the running time of our method for some examples. It illustrates the general relation between the total time and the time spent in solving the IP. The reported times are averages of 200 runs.

Our Python implementation is not optimal but versatile. It generates the IP problem file and call GLPK [19] (in particular the utility `glpsol`) to solve the IP. QHull [6] is also called as an external utility and has a negligible running time. Better timings should be expected from a production-ready C++ implementation with direct linking to GLPK and QHull libraries.

	Total	**IP**	**Other**		**Total**	**IP**	**Other**
Fig. 9a	133	14	119	Fig. 9b	142	19	123
Fig. 6b	55	12	43	Fig. 6c	55	12	43
Fig. 7b-top	16	4	12	Fig. 7c-top	24	4	20
Fig. 7b-mid	29	6	23	Fig. 7c-mid	37	7	30
Fig. 7b-bot	44	7	37	Fig. 7c-bot	51	6	45

Table 2: Running time (in milliseconds) of our implementation. Columns: **Total** – total running time, **IP** – time for finding the IP solution, **Other** – rest of the time.

6 Further simplifications of the scaffold

As we discussed in Section 3, the minimal number of points for each cell can be set to three. This decreases the number of quads in the final scaffold and might generate triangular cross-sections along some skeleton edges. In Figure 14 we recompute the example in Figure 1a with the new lower bounds, this decreases the number of quads in the standard scaffold from 192 to 144.

Table 3 summarizes the number of quads and vertices per valency of some of the scaffolds in this paper, all of which are optimally computed with our implementation. It shows that the symmetric scaffolds on Figure 4 and 6 are also optimal standard scaffolds.

	$\mathbf{v_3}$	$\mathbf{v_4}$	$\mathbf{v_6}$	$\mathbf{t_{quads}}$		$\mathbf{v_3}$	$\mathbf{v_4}$	$\mathbf{v_6}$	$\mathbf{t_{quads}}$
Fig. 14	*	*	96	144	Fig. 6b	18	33	14	63
Fig. 1a	*	48	96	192	Fig. 6c	18	33	14	63
Fig. 1b	*	120	*	120	Fig. 16b	48	66	12	108
Fig. 1c	*	72	*	72	Fig. 16c	96	150	12	216
Fig. 1d	*	72	*	72	Fig. 16d	192	318	12	432
Fig. 4a-b	24	8	4	24	Fig. 16e	384	654	12	864

Table 3: Summary of some scaffolds in the paper. Columns: $\mathbf{v_k}$ – number of vertices with valency $\mathbf{k}$, $\mathbf{t_{quads}}$ – total number of quads. The entries $*$ mean zero.

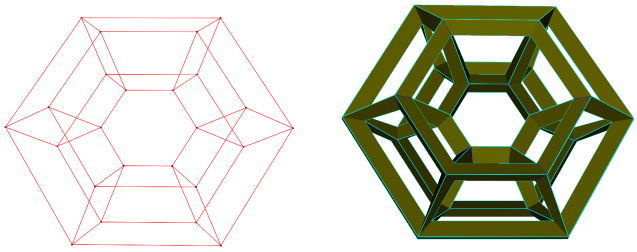
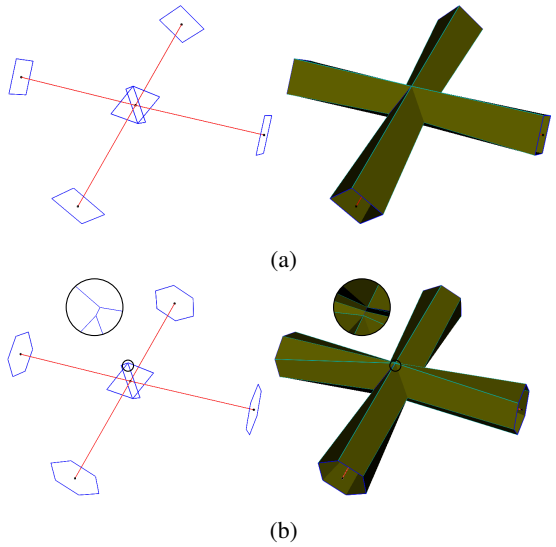


Figure 14: Example of a scaffold constructed with 3 as minimal number of points on each cell. Compare with Figure 1a.

Another simplification arises by noticing that any partition of the sphere with an *inscribable* dual gives feasible solutions for the IPs. Thus the computation of Voronoi regions can be modified by changing the minimal angle for which two triangles sharing an edge in

the convex hull are considered to be coplanar. In our experience, this helps to reduce the complexity of the cells around joints and in general reduces (even more) the number of quads needed to construct a scaffold. As an example, in Figure 15 four close-to-coplanar points can be considered coplanar, yielding compatible cells with less points (hence less quads). On the other hand, due to round-off errors coplanar points may appear as not coplanar. This issue provokes that very small arcs appear on the boundary of Voronoi regions, consequently some quads may look like triangles at a large scale (Figure 15b).



(a)

(b)

Figure 15: Close-to-coplanar points can be treated as coplanar as a way to improve the scaffold. In the picture the four intersection points in the joint are not coplanar. In (a) they are treated as coplanar points, in (b) the exact Voronoi regions (and cells) are computed.

7 Application to polygonization

In this paper we focus on the feasibility of the scaffold construction, including the symmetric and regular variants. The scaffold is in most applications an *intermediate* step [4, 5, 7, 15, 16, 23, 24, 26]. Although we present theoretical results and practical algorithms we also want to showcase an example application.

Polygonization of implicit surfaces is usually done with Marching Cubes algorithm [18] or one of its variants [25, Chapter 2]. For computer graphics applications quad-dominant meshes are more desirable and this requires a re-meshing of the standard triangular meshes obtained with the Marching Cubes algorithms [9, Section 6.6]. For articulated models, it is a good improvement if the quad mesh follows the structure of the skeleton. A scaffolding technique can be used to bypass the re-meshing process and get directly a quad-dominant mesh that follows the skeleton.

Figure 16 shows a mesh computed with our scaffolding technique. The implicit surface (Figure 16a) is a convolution surface [27] for which a scaffold is computed with a higher number of subdivisions for each arc: multiplying the standard solution by an integer $\kappa > 0$ that also defines the number of quads along a segment. To obtain the surface mesh we project the scaffold quads onto the surface. This is done by ray-shooting: compute the intersection with the surface of rays emanating from the skeleton in the direction given by quad vertices. A similar projection was done in [1], the main difference with our approach is that the mesh we project (scaffold) is computed for the whole skeleton while in [1] a fixed set of meshes is defined for each line segment. At the tips of dangling nodes, the quads are computed by creating a polar-annular region with a singular point (as done in [4]). The polar-annular meshes transform all valency 3 vertices on the boundary of the scaffold mesh into valency 4 vertices at the cost of having an extra high valency vertex per dangling node.

8 Conclusions

We presented a method to construct a coarse quad-dominant mesh around a skeleton made of line segments that can serve as *intermediate* step in several applications. The mesh follows the structure of the skeleton and is computed in an optimal way, minimizing the total number of quads. Our method works for any skeleton even in the presence of cycles. We modeled the problem as an Integer Linear Program and proved its feasibility. We presented variants of our method: we can generate a mesh with the same number of quads around each line segment, or a mesh with the same symmetries as the underlying skeleton. It is also possible to satisfy both conditions at the same time. The scaffold thus obtained can be used to polygonize implicit surfaces around the skeleton or as an intermediate step in other applications. Our method overcomes some limitations of, and add extra features to, previous work. The algorithmic description of each step gives a basis on which multiple implementations can be developed.

References

[1] A. Angelidis and M.-P. Cani. Adaptive implicit modeling using subdivision curves and surfaces as skeletons. pages 45–52, 2002.

[2] J. M. Augenbaum and C. S. Peskin. On the construction of the voronoi mesh on a sphere. *Journal of Computational Physics*, 59(2):177–192, 1985.

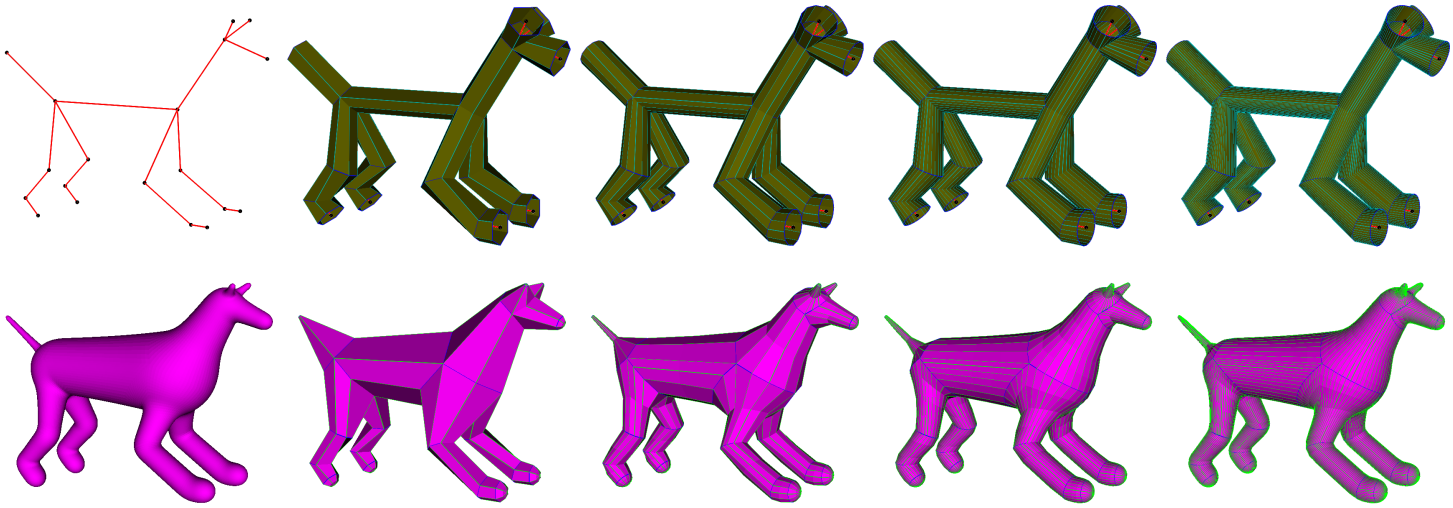
[3] F. Aurenhammer. Voronoi diagrams – a survey of a fundamental geometric data structure. *ACM Comput. Surv.*, 23(3):345–405, 1991.

[4] J. Bærentzen, M. Misztal, and K. Wełnicka. Converting skeletal structures to quad dominant meshes. *Computers & Graphics*, 36(5):555–561, 2012. Shape Modeling International (SMI) Conference 2012.

[5] J. A. Bærentzen, R. Abdrashitov, and K. Singh. Interactive shape modeling using a skeleton-mesh co-representation. *ACM Trans. Graph.*, 33(4):132:1–132:10, 2014.

[6] C. B. Barber, D. P. Dobkin, and H. Huhdanpaa. The quickhull algorithm for convex hulls. *ACM Trans. Math. Softw.*, 22(4):469–483, 1996.

[7] A. Blidia, B. Mourrain, and N. Villamizar. G1-smooth splines on quad meshes with 4-split macro-patch elements. *Computer Aided Geometric Design*, 52-53:106–125, 2017.



(a) The model. (b) Standard ($\kappa = 1$). (c) $\kappa = 2$. (d) $\kappa = 4$. (e) $\kappa = 8$.

Figure 16: The scaffold method as a basis for a quad dominant mesh of an implicit surface. (a) Top: the skeleton, bottom: an implicit convolution surface around the skeleton. The rest of the columns (b-e) represent the standard scaffold solution (of subdivision of arcs for compatible cells) multiplied by the constant κ. In (b-e) κ also defines the number of quads along each line segment.

[8] J. Bloomenthal and K. Shoemake. Convolution surfaces. *ACM SIGGRAPH Computer Graphics*, 25(4):251–256, 1991.

[9] M. Botsch, L. Kobbelt, M. Pauly, P. Alliez, and B. Levy. *Polygon Mesh Processing*. A K Peters/CRC Press, 2010.

[10] K. Q. Brown. *Geometric Transforms for Fast Geometric Algorithms*. Phd thesis, Carnegie-Mellon University, Pittsburgh, Penn., 1979.

[11] D.-S. Chen, R. G. Batson, and Y. Dang. *Applied Integer Programming*. John Wiley & Sons, Inc., Hoboken, NJ, USA, 2009.

[12] M. B. Dillencourt and W. D. Smith. Graph-theoretical conditions for inscribability and delaunay realizability. *Discrete Mathematics*, 161(1):63–77, 1996.

[13] A. J. Fuentes Suárez and E. Hubert. Scaffolding skeletons using spherical voronoi diagrams. *Electronic Notes in Discrete Mathematics*, 62:45–50, 2017. LAGOS'17 IX Latin and American Algorithms, Graphs and Optimization.

[14] C. I. Grima and A. Márquez. *Computational Geometry on Surfaces*. Springer Netherlands, Dordrecht, 2001.

[15] Z. Ji, L. Liu, and Y. Wang. Bmesh: A modeling system for base meshes of 3d articulated shapes. *Computer Graphics Forum*, 29(7):2169–2177, 2010.

[16] K. Karčiauskas and J. Peters. Curvature continuous bi-4 constructions for scaffold- and sphere-like surfaces. *CAD Computer Aided Design*, 78:48–59, 2016.

[17] R. M. Karp. *Reducibility Among Combinatorial Problems*, pages 219–241. Springer Berlin Heidelberg, Berlin, Heidelberg, 2010.

[18] W. E. Lorensen and H. E. Cline. Marching cubes: A high resolution 3d surface construction algorithm. *Proceedings of the 14th annual conference on Computer graphics and interactive techniques - SIGGRAPH '87*, 21(4):163–169, 1987.

[19] A. Makhorin. *GNU Linear Programming Kit - Reference Manual*, volume 1. 2003. https://www.gnu.org/software/glpk/.

[20] H.-S. Na, C.-N. Lee, and O. Cheong. Voronoi diagrams on the sphere. *Computational Geometry*, 23(2):183–194, 2002.

[21] A. Panotopoulou, E. Ross, K. Welker, E. Hubert, and G. Morin. Scaffolding a skeleton. Preprint, 2017.

[22] I. Rivin. A characterization of ideal polyhedra in hyperbolic 3-space. *Annals of Mathematics*, 143:51–70, 1996.

[23] V. Srinivasan, E. Mandal, and E. Akleman. Solidifying wireframes. In *Bridges: Mathematical Connections in Art, Music, and Science 2004*, Banf, 2005.

[24] F. Usai, M. Livesu, E. Puppo, M. Tarini, and R. Scateni. Extraction of the quad layout of a triangle mesh guided by its curve skeleton. *ACM Transactions on Graphics*, 35(1):1–13, 2015.

[25] R. Wenger. *Isosurfaces. Geometry, Topology & Algortihms*. CRC Press, 2013.

[26] C.-Y. Yao, H.-K. Chu, T. Ju, and T.-Y. Lee. Compatible quadrangulation by sketching. *Computer Animation and Virtual Worlds*, 20(2-3):101–109, 2009.

[27] C. Zanni. *Skeleton-based Implicit Modeling & Applications*. Phd thesis, Université de Grenoble, Grenoble, 2013.

Acknowledgments: This project has received funding from the European Union's Horizon 2020 research and innovation programme under the Marie Skłodowska-Curie grant agreement No 675789.

Marie Skłodowska-Curie Actions