A direct slicing method for optimized lattice structures

Gilles Foucault, Frédéric Vignat, Pierre-Thomas Doutre

Mechanics & Industry **25**, 26 (2024)
© G. Foucault et al., Published by EDP Sciences, 2024
https://doi.org/10.1051/meca/2024019

Mechanics & Industry

Available online at:
www.mechanics-industry.org

ORIGINAL ARTICLE OPEN ACCESS

A direct slicing method for optimized lattice structures

Gilles Foucault*, Frederic Vignat, and Pierre-Thomas Doutre

Univ. Grenoble Alpes, CNRS, Grenoble Institute of Engineering Univ. Grenoble Alpes, G-SCOP, 38000 Grenoble, France

Received: 19 October 2023 / Accepted: 11 July 2024

Abstract. Previous research show that conventional CAD B-Rep methods are highly inefficient to design large lattice structures composed of several thousands of geometric primitives describing struts [2,3]. This paper aims at proposing a method for slicing directly a lattice structure represented as a skeleton: a set of struts connecting nodes, where radius can be constant (cylinder) or linear (cones). The contribution of this method is twofold (1) efficiency: the CAD B-Rep representation is replaced with the skeleton representation which is robust and requires very low resources to be generated. (2) Accuracy: The method generates the explicit and exact geometry of layers with analytic curves, enabling the export into layers with polygons having an arbitrary accuracy.

Keywords: Additive manufacturing / slicing / lattice structures / Digital Chain

1 Introduction

Using additive manufacturing technologies, complex geometrical entities, particularly lattice structures, can be created [1]. They enable significant weight reduction in products [1]. The use of lattice structures can therefore be strategic in lightweight applications. However, they are rarely used today. Indeed Conventional CAD B-Rep methods are highly inefficient to design large lattice structures composed of several thousands of geometric primitives describing struts [2,3]. This type of representation is necessary in conventional parts for their visualization and finite element analysis (FEA) using 3D elements. However, this approach becomes computationally expensive for lattice structures due to their intricate nature. Thankfully, FEA can analyze these structures using beam elements instead of full volume models. Beam element simulations are significantly faster and sufficiently accurate for structures with less than 5% density [1]. Therefore, creating a complete volume model becomes unnecessary for analyzing lattice structures.

The slicing process converts the 3D shape into 2D horizontal layers of material to be deposited. The layers are then transferred to the additive manufacturing machine which generates its own build instructions.

In the current practice, the 3D shape model is created using a CAD modeler, and converted into an intermediate STL format which approximates the model with a triangulation [4–6]. The slicer process intersects the STL file with successive layers' planes to generate 2D slices. However, building the CAD model of lattice structures composed of a large number of struts is resource consuming due to the large number of surfaces [2,7].

Several works proposed improvement in the lattice structure model creation.

One can classify these approaches as following:

- Triangulation approaches which generate a triangulation from a lattice truss representation.
- Direct slicing methods which generate layers' 2D geometry directly from a lattice truss representation.

The slicing step can then be performed using algorithms such as Slic3r proposed in OpenSource [8]. First, a list of Z coordinates containing the height of each slice is generated. Several slicing processes are executed in parallel at different heights. The slicing process consists in computing the intersection lines between STL triangles and the slicing plane. Pair of lines belonging to adjacent triangles are connected at their common vertex, hence forming a line-vertex adjacency graph. The final slicing step consists in forming the boundary loops of the slice by extracting cycles in the adjacency graph.

Marching Cubes Method (MCM) [9] tends to be more and more used to triangulate lattice structures for additive manufacturing. This technique consists in using voxelization of the 3D space around an object as well as a scalar distance field defined on each node of the voxel grid. Then, it uses simple rules to tessellate the wanted iso-value of the distance field. The distance field can be deduced from a skeleton. This method generates a large number of triangles which are usually of irregular geometric quality throughout

* e-mail: gilles.foucault@univ-grenoble-alpes.fr

the mesh. This is due to the fact that the number of triangles and their shape quality are directly linked to the size of the voxelization grid as well as to the orientation of the voxel grid relatively to the object. This is a strong limitation when considering lattice structures made of beams with widely varying directions.

The triangulation approach proposed in [10] generates the triangulation on each cylinders of every beam of the skeleton. A special Boolean operator moves vertices located in the interior of the lattice along their underlying cylindrical surface to reach the intersection curve with the closest adjacent cylinder. This method results in a watertight and manifold triangulation, while minimizing the number of triangles and respecting a chordal error limit. However, the method would be complex to extend to lattice structures featuring conical struts, adjacent struts featuring different sizes representing graded architecture materials.

More recently, Savio [11] used mesh subdivision techniques to triangulate lattice structures by generating on each strut two truncated square pyramids with respect to a predefined strut size. Each truncated pyramid is composed of 4 quadrangles for the side faces. The process ends with the Catmull-Clark mesh subdivision, resulting in a smooth triangulation of the lattice, featuring rounded struts' sections, variable radiuses, and blends at nodes. This method has two major limitations: the result contains a very large number of triangles, and the size of blends joining struts' surfaces cannot be controlled.

Direct slicing approaches generate slices directly from an implicit lattice representation such as CSG tree [12], layered depth-normal images (LDNI) [13], avoiding the need to create a watertight triangulation.

A direct slicing method for NURBS surfaces was proposed in [14]. The method is based on a ray-casting technique. Rays are casted in the plane of the layer, intersection points with the surface are stored, along with their entry/exit status and their surface's normal. The slice geometry is then generated by connecting neighbor points, based on normal directions and points status. The method can be applied to weak B-Rep definitions, such as non-watertight solids, or overlapping surfaces. The major limitation of this approach is the need for a CAD B-Rep input, for which CAD modelers are highly inefficient to generate.

A Ray-tracing direct slicing method was proposed to generate bitmap images of slices directly from the mathematical description of CSG primitives representing the lattice structure [12]. The limitation of this approach is the pixelation of bitmaps which requires significant resolution to represent the layers with the accuracy requirements.

Quador beams [15] are surfaces that generalize the straight (cone) beams to quadric surface of revolution (revolution of line, hyperbola, parabola, ellipse). Quador have both implicit and parameterized representation [16]. To create a direct slicer for lattice structures composed with Quador beams, authors in [17] proposed a ray-casting algorithm to generate the plane-strut intersection ellipses of the lattice slices. This method generates slices with the accuracy defined by the resolution of the slice image.

The algorithm presented in this article can be summarized as follows. The first step is the explicit definition of geometry parameters of plain cylinders, cones and cap spheres attached to the input skeleton format. Then, the tentative curves generated by the intersection between the geometry and the slicing plane are trimmed in order to only retain subsets lying on the boundary of the lattice 3D domain. The results represent the layer of the slice with a set of closed contours, each of which consisting in the sequence of parametric curves representing the boundary domain of the layer.

Our specific contribution in this article is a slicing method acting directly on skeleton models representing a lattice structure as a graph of struts and nodes [2].

The proposed slicer algorithm bypasses STL file generation. Unlike other direct slicing approaches relying on ray-casting methods with an accuracy relative to the grid resolution, our method relies on a CAD kernel and generates the parametric curves describing the exact geometry of the lattice slices, based on intersection curves between parameterized representation of strut surfaces (cones, cylinders) and the slicing plane. The paper is organized as follows. Section 2 defines the geometric model of the lattice structure. Section 3 presents an overview of the proposed approach for the slicing procedure. Section 4 presents the 3D definition of independent cones, cylinders, and spheres attached to lattice's struts. Section 5 presents the direct slicing method that generates the set of tentative curves, and the first step that discards struts lying outside the slicing plane. Section 6 presents the trimming of interior subsets in candidate curves, producing the slice boundary on cones or cylinders. Section 7 presents the plane intersection of cap spheres that produces the slice boundary lying on spheres. For a quick overview, the graphical abstract in Appendix A.1 presents graphically the whole process.

2 Geometric model for lattice structures

Optimizing the material usage in lattice structures leads to configurations where the relative density is variable throughout the whole design space. The dimension of each elementary cell and the radius of each strut can vary throughout the design space to produce the optimized lattice structure. Indeed, the generation of this truss structure relies on a skeleton model, as described in [3]. This skeleton model entails a set of point coordinates corresponding to each vertex of the truss structure. Parameters thus include the coordinates of each node, the connectivity between each node (describing how they are connected), as well as a radius assigned to each node. To fill a cube with truss structures of variable density, the dimension of each elementary cell remains constant, with only the beam size varying. However, to fill a cylindrical geometry with variable density, it is necessary to slightly adjust the dimension and geometry of the unit cell to fit this specific geometry, as well as vary the beam size.

This optimized lattice structure generates a skeleton model, which is the input data of the slicer.

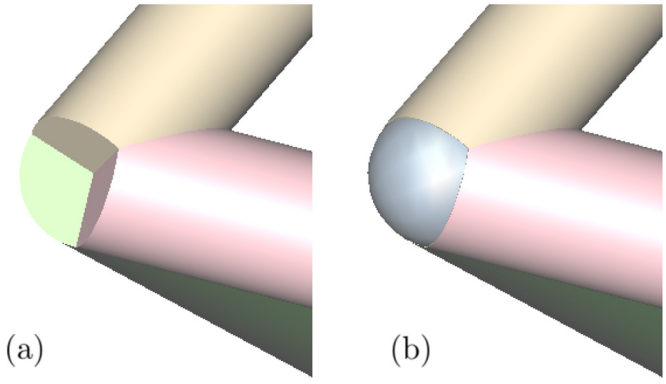

Fig. 1. (a) The material discontinuity located at the intersection of adjacent struts (b) Discontinuity avoided by the insertion of cap spheres on nodes.

The skeleton model is decomposed into a set of *nodes* and a set of *struts* linking them. Each node has two or more adjacent struts. Struts' cone shapes, tangent to node spheres, represent the linearly changing size of struts between nodes. The addition of cap spheres at nodes not only smoothens and simplifies the geometry, but also avoids the presence of stress raisers (see Fig. 1).

The graph $G=(V,E)$, represents the lattice topology where V is the set of vertices and E is the set of edges, each of which representing a strut $E_{i,j}$ connecting vertices V_i and V_j. Each vertex V_i represents a sphere Ω_{Vi} centered on point $\boldsymbol{P}_i$ and featuring radius R_i. Each edge $E_{i,j}$ represents a strut with $\Omega_{\mathrm{Ei,j}}$ being either a cylinder if $R_j=R_i$, or a finite-length cone tangent to spheres Ω_{Vi}, Ω_{Vj} otherwise (see Fig. 2). The lattice volume Ω_L is the union of spheres Ω_{Vi} and cones $\Omega_{\mathrm{Ei,j}}$ (see Fig. 2):

$$\Omega_L = \left(\underset{i}{\cup}\Omega_{Vi}\right)\cup\left(\underset{i,j}{\cup}\Omega_{Ei,j}\right)$$

$$\forall V_i\in V(G);\forall E_{i,j}\in E(G).$$

This graph-based data structure is computationally efficient to describe lattice structures having hundreds of thousands of struts, and is adapted to develop direct slicing algorithm.

The direct slicing algorithm works with an exact, non-triangulated, representation of the solid geometry, and converts directly each 3D primitive into 2D section slices containing exact slice curves.

The terms adjacency, connected, neighborhood, incident refer to usual definitions given in the Graph Theory [18].

3 The proposed approach

The slicing direction is chosen as the Z-direction, and each slice represents the geometry of the layer printed at a given Z coordinate. A slice of the solid Ω_{L} with the slicing plane Π is the planar point set resulting from the Boolean intersection $\Omega_{\mathrm{L}}\cap\Pi$. Each slice must be represented by the set of oriented curves representing the boundary between interior and exterior domains of the slice, forming closed loops and oriented in the counter-clockwise sense. The slicing method consists in evaluating the boundary of the intersection, i.e. $C^{sli}=\partial(\Omega_{\mathrm{L}}\cap\Pi)$.

The slice boundary can be computed by the following algorithm, which is an instance of the familiar generate-and-test paradigm in Computer Science:

– Generate a sufficient set of tentative curves $\{C^t\}\supset C^{sli}$.
– For each C^t do.
– Discard segments of C^t which are not on $\partial\Omega_L$.

$$C^{sli}=C^{sli}\cup\left(C^t\subset\partial\Omega_L\right).$$

Sufficient sets of tentative curves can be generated by using the basic fact $\partial(A\cup B)\subset(\partial A\cup\partial B)$. Slice boundary is hence a subset of primitives' slice boundary $\partial\Omega\cap\Pi$:

$$\left\{C^t\right\}=\underset{i}{\cup}(\partial\Omega_{Vi}\cap\Pi)\cup\underset{i,j}{\cup}\left(\partial\Omega_{Ei,j}\cap\Pi\right).$$

The following method evaluates the slice using above property:

Let $S_{Vi}=\partial\Omega_{Vi}$ be the boundary surface of Ω_{Vi}, $S_{Ei,j}=\Omega_{Ei,j}$ be the boundary surface of $\Omega_{Ei,j}$, and Π the slicing plane.

– Generate sets of tentative curves on cones by intersecting cone surfaces $S_{Ei,j}$ with slicing plane Π (see Fig. 3a)

$$\left\{C^t_{Ei,j}\right\}=S_{Ei,j}\cap\Pi\forall E_{i,j}\in E(G).$$

– Discard segments from $\left\{C^t_{Ei,j}\right\}$ which are inside the neighbor cones $\Omega_{Ek,l}$ (see Fig. 3b):

$$\{C_{Ei,j}\}=\left\{C^t_{Ei,j}\right\}\setminus\Omega_{Ek,l}$$

$$\forall E_{k,l}\in N\left(E_{i,j}\right)$$

Where $A\setminus B$ is the set of points in A but not in B.

– Generate sets of tentative curves on spheres by intersecting sphere surfaces S_{Vi} with plane Π(see Fig. 3a)

$$\left\{C^t_{Vi}\right\}=S_{Vi}\cap\Pi\forall V_i\in V(G).$$

– Discard segments from $\left\{C^t_{Vi}\right\}$ which are inside the cones $\Omega_{Ei,j}$ incident to sphere Ω_{Vi} (see Fig. 3b):

$$\{C_{Vi}\}=\left\{C^t_{Vi}\right\}\setminus\Omega_{Ei,j}\forall E_{i,j}\in E(G)$$

where $A\setminus B$ is the set of points in A but not in$B$.

4 Sphere and cone primitives

The shape of strut $E_{i,j}$ is the Boolean union of three solid primitives: one finite-length cone $\Omega_{Ei,j}$ (see Fig. 4a), and two spheres Ω_{Vi} and Ω_{Vj} of radiuses R_i and R_j centered on points $\boldsymbol{P}_i$ and $\boldsymbol{P}_j$ (see Fig. 4b). The cone's side surface $S_{Ei,j}$ is tangent to spheres S_{Vi} and S_{Vj} along circles $C^b_{i,j}$ and $C^c_{i,j}$ (Fig. 4c).

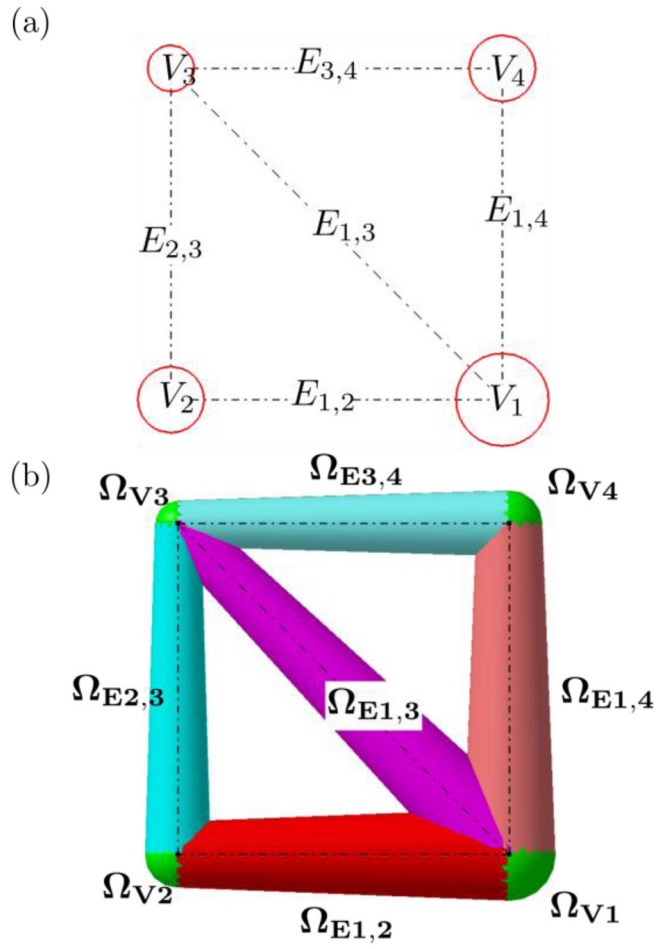

Fig. 2. The graph representation for a lattice structure, (a) lattice graph with vertices Vi and Vj and edges $E_{i,j}$, (b) sphere Ω_{Vi} and cones $\Omega_{Ei,j}$ primitives

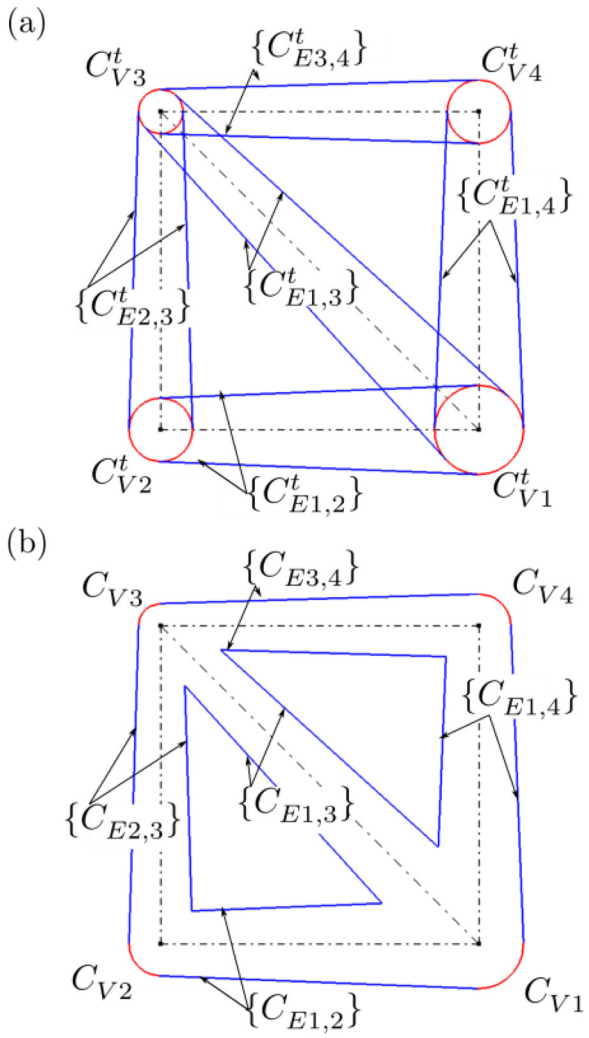

Fig. 3. (a) nodes and strut slices, (b) lattice boundary slices.

The finite-length cone is defined by its base point, cap point, base radius, and cap radius (see Figs. 5 and 6).

The following equations present struts' geometry completely derived from lattice vertices attributes ($\boldsymbol{P}_i$, R_i).

The semi-angle $\alpha_{i,j}$ of the cone $S_{Ei,j}$ is:

$$\alpha_{i,j} = \text{asin}\left(\frac{R_j - R_i}{|| \quad \boldsymbol{P_iP_j} \quad ||}\right)$$

where $\boldsymbol{P_iP_j} = \boldsymbol{P_j} - \boldsymbol{P_i}$.

Base and cap circles are normal to $\boldsymbol{P_iP_j}$.

Base circle parameters:

– Radius $R^b_{i,j} = R_i \cdot \cos\alpha_{i,j}$

– Center $\boldsymbol{P^b_{ij}} = \boldsymbol{P_i} - R_i \cdot \frac{\boldsymbol{P_iP_j}}{||\boldsymbol{P_iP_j}||} \sin\alpha_{i,j}$.

Cap circle parameters:

– Radius $R^c_{i,j} = R_j \cdot \cos\alpha_{i,j}$

– Center $\boldsymbol{P^c_{ij}} = \boldsymbol{P_j} - R_j \cdot \frac{\boldsymbol{P_iP_j}}{||\boldsymbol{P_iP_j}||} \sin\alpha_{i,j}$.

Algorithm 1:

Create cone ($\boldsymbol{P_i}$,$\boldsymbol{R_i}$,$\boldsymbol{P_j}$,$\boldsymbol{R_j}$)

Return value: cone $S_{E_{i,j}}$ and circles $C^b_{i,j}$;$C^c_{i,j}$

Axis' direction: $\boldsymbol{P_iP_j} = \boldsymbol{P_j} - \boldsymbol{P_i}$

Cone Semi-angle: $\alpha_{i,j} = \text{asin}\left(\frac{R_j - R_i}{\boldsymbol{P_iP_j}}\right)$

Create cone surface:

$$S_{E_{i,j}} = Cone\left(\boldsymbol{P^b_{ij}}, \boldsymbol{P_iP_j}, R^b_{i,j}, \alpha_{i,j}\right)$$

Create cone base circle $C^b_{i,j}$:

$$center = \boldsymbol{P^b_{ij}} = \boldsymbol{P_i} - R_i \cdot \frac{\boldsymbol{P_iP_j}}{||\boldsymbol{P_iP_j}||} \sin\alpha_{i,j}$$

$$radius = R^b_{i,j} = R_i \cdot \cos\alpha_{i,j}$$

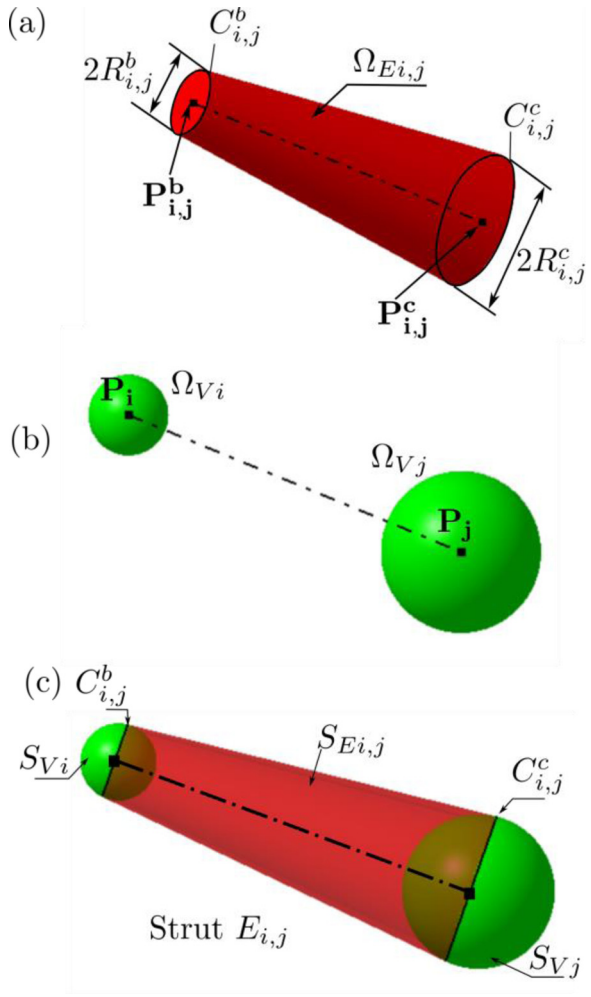

Fig. 4. geometry of a strut $E_{i,j}$.

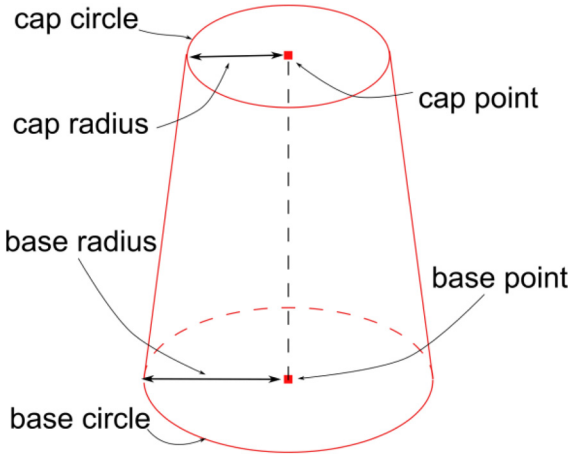

Fig. 5. Cone primitive definition.

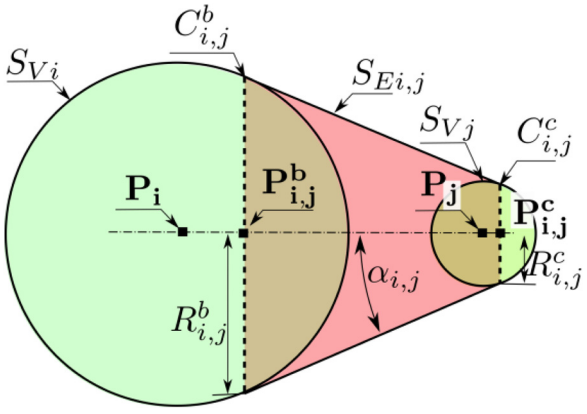

Fig. 6. Cone parameters.

$$normal = \boldsymbol{P_i P_j}$$

$$C^b_{i,j} = Circle(center, radius, normal)$$

Create cone cap circle $C^c_{i,j}$:

$$center = \boldsymbol{P^c_{ij}} = \boldsymbol{P_j} - R_j \cdot \frac{\boldsymbol{P_i P_j}}{||\boldsymbol{P_i P_j}||} \sin\,\alpha_{i,j}$$

$$radius = R^c_{i,j} = R_j \cdot \cos\,\alpha_{i,j}$$

$$normal = \boldsymbol{P_i P_j}$$

$$C^c_{i,j} = Circle(center, radius, normal)$$

5 The proposed direct slicing method

The slicing works with four geometric steps:

- **Z-Buffering**: (See step 1 in Fig. 7) Z-buffers are lists of struts intersected by each slice section plane Π_i defined as $Z_i = z_0 + i \cdot \Delta z \,\forall\, i \in [0\,;\, N-1]$.
- **Slicing procedure on cones/cylinders**: (See step 2 in Fig. 7) The set of intersected struts contained in the slice Z-Buffer is retrieved. The cone and cylinder surfaces are sliced. The result is a set of edges representing the exact geometry of struts' section.
- **Trimming of interior edges**: (See step 3 in Fig. 7) for each slice curve, the slices of adjacent struts are merged by intersecting/splitting edges, and removing interior subsets.
- **Slicing procedure on cap spheres**: (See step 4 in Fig. 7) cap sphere subdomains lying on the outside of adjacent struts are sliced, forming arcs of circles.

5.1 Rejection test and Z-buffering (step 1)

The lattice domain is explored along Z axis to discard struts that are guaranteed not to intersect the slicing plane, thus minimizing the number of intersection tests (see Fig. 8). The height of the first layer is $Z = Z_0$, and next section planes are equally spaced by the ΔZ layer thickness. Hence, the height of the k^{th} section plane designated Π_k is $Z_k = Z_0 + k.\,\Delta Z$.

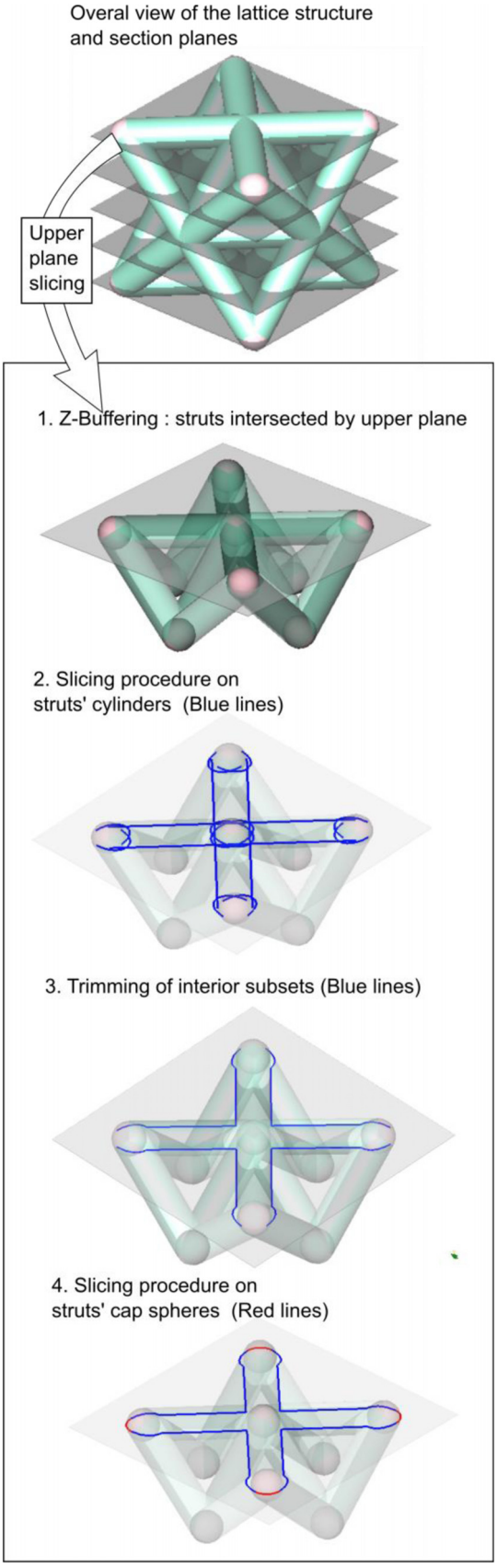

Fig. 7. The four slicing steps.

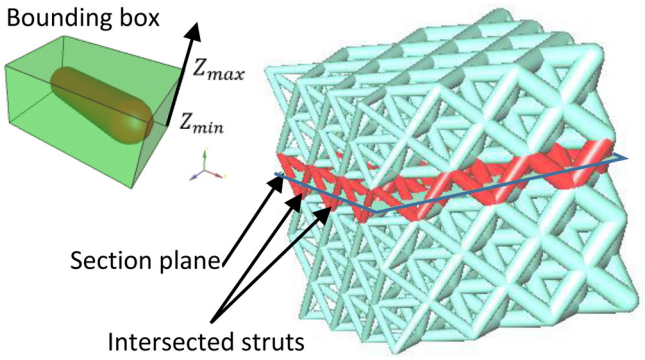

Fig. 8. Z-buffer: (left) Bounding box of a strut, (right) struts intersected by the plane are colored in red.

Each Z-buffer refers to the kth slice, section plane Π_k, and the list of intersected struts. The Z-buffering is obtained using min-max coordinates of struts along Z axis.

Given a strut $\Omega_{Ei,j}$, its min-max coordinates along Z axis are:

$$Z_{min}(\Omega_{Ei,j}) = min(\boldsymbol{P_i}\cdot\boldsymbol{z} - R_i; \boldsymbol{P_j}\cdot\boldsymbol{z} - R_j)$$

$$Z_{max}(\Omega_{Ei,j}) = \max(\boldsymbol{P_i}\cdot\boldsymbol{z} + R_i; \boldsymbol{P_j}\cdot\boldsymbol{z} + R_j).$$

Strut $\Omega_{Ei,j}$ is cut by plane Π_k if

$$k \in \left[\sup\left(\frac{Z_{min}(\Omega_{Ei,j}) - Z_0}{\Delta Z}\right); \inf\left(\frac{Z_{max}(\Omega_{Eij}) - Z_0}{\Delta Z}\right)\right].$$

6 Slicing of cones/cylinders (step 2)

For all struts cut by plane Π_k, contained in Z-Buffer at index k, their primitives are intersected with plane Π_k to generate slice curves. The process is decomposed in two independent tasks: the slicing of strut surfaces $S_{Ei,j}$, and the slicing of spheres S_{Vi}.

6.1 Slicing of strut surfaces (cone or cylinder)

Surfaces intersections $S_{Ei,j} \cap \Pi_k$ generate a set of tentative curves $\{C^t_{Ei,j}\}$ containing the slice boundary. Tentative curves are either circle, ellipse, line, parabola, or hyperbola. To filter the curve' subset lying on the lattice boundary, the process splits curve $C^t_{Ei,j}$ at struts' end circles $C^b_{i,j}$ and $C^c_{i,j}$ (see Figs. 9 and 10), and discards segments lying inside neighbor struts $N(E_{i,j})$, resulting in curves $\{C_{Ei,j}\}$:

- The tentative curve $C_{Ei,j}$ is split at points $\left\{P^b_{i,j}\right\} = \Pi_k \cap C^b_{i,j}$ and $\left\{P^c_{i,j}\right\} = \Pi_k \cap C^c_{i,j}$ (see Fig. 9), resulting in $\{C^t_{Ei,j,l}\}$.
- Each curve $\in C^t_{Ei,j,l}$ lie entirely either in the surface $S_{Ei,j}$, or in the exterior domain of $\Omega_{Ei,j}$.
- By pruning middle point of tentative curves' segments $Midpoint\left(C^t_{Ei,j,l}\right)$, exterior segments are discarded, and only subsets lying entirely on surface $S_{Ei,j}$ remain.

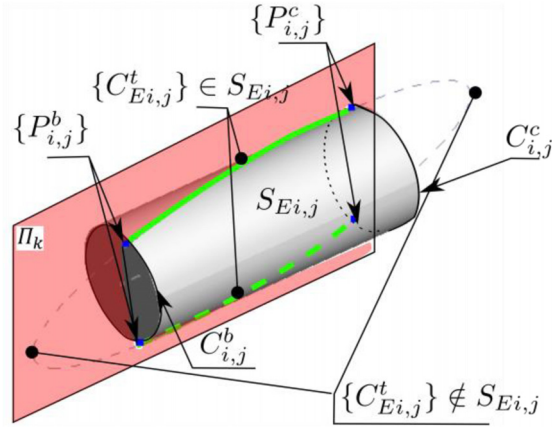

Fig. 9. Slicing of strut surface $S_{Ei,j}$: ellipse trimmed with $C^b_{i,j}$ and $C^c_{i,j}$ strut's bounding circles.

Algorithm 2:

Slice strut's surface $\boldsymbol{S_{Ei,j}}$

Input: Π_k, $\boldsymbol{P_i}$, R_i, $\boldsymbol{P_j}$, R_j
Return value: curve set $\{C^t_{Ei,j}\} = S_{Ei,j} \cap \Pi_k$
If $(R_i = R_j = R)$
$S_{E_{i,j}}$ = cylinder(radius R, axis $[\boldsymbol{P_i P_j}]$) bounded by circles:

$$C^b_{i,j} = circle(\boldsymbol{P_i}, R, \boldsymbol{P_i P_j})$$
$$C^c_{i,j} = circle(\boldsymbol{P_j}, R, \boldsymbol{P_i P_j})$$

Set points $\left\{\boldsymbol{P^b_{i,j}}\right\} = \Pi_k \cap C^b_{i,j}$ and $\left\{\boldsymbol{P^c_{i,j}}\right\} = \Pi_k \cap C^c_{i,j}$,

If $[\boldsymbol{P_i P_j}]$ is parallel to Π_k
$\{C^t_{Ei,j}\}$ = pair of segments on cylinder generatrixes $\left[\boldsymbol{P^b_{i,j,1}}\boldsymbol{P^c_{i,j,1}}\right]$ and $\left[\boldsymbol{P^b_{i,j,2}}\boldsymbol{P^c_{i,j,2}}\right]$
Exit function
Else

$$C_i = \Pi_k \cap S_{ci} \text{ (ellipse)}$$

Else
Generate Cone and circles $S_{E_{i,j}}$; $C^b_{i,j}$; $C^c_{i,j}$ (see Algorithm 1)
$\{C^t_{Ei,j}\} = \Pi_k \cap S_{Ei,j}$ (=Create set of tentative curves on cone or cylinder $S_{E_{i,j}}$)

$\left\{C^t_{Ei,j,m}\right\}$ = Subsets of $\{C^t_{Ei,j}\}$ split at intersection points $\left\{\boldsymbol{P^b_{i,j}}\right\}$, $\left\{\boldsymbol{P^c_{i,j}}\right\}$
Discard elements of $\left\{C^t_{Ei,j,m}\right\}$ where $Midpoint\left(C^t_{Ei,j,m}\right) \notin S_{Ei,j}$

Fig. 10. Strut surface slicing algorithm.

7 Trimming of interior subsets (step 3)

The step 3 of the slicing algorithm consists in discarding subsets of tentative curves $\{C^t_{Ei,j}\}$ lying in interior of intersecting struts $\Omega_{Ek,l}$ such as $\{C^t_{Ei,j}\} \cap \{\Omega_{Ek,l}\} \neq \emptyset$ (see Figs. 5.4 and 11).

A necessary condition for a strut $E_{k,l}$ to intersect one tentative curve $\{C^t_{Ei,j}\}$ is the fact that his volume $\Omega_{Ek,l}$ intersects $\Omega_{Ei,j}$. Also, the boundary of the subdomain of $\{C^t_{Ei,j}\}$ lying inside the intersecting strut $\Omega_{Ek,l}$ is located on the boundary surface $S_{Ek,l}$ and the slicing plane Π_k, then on the tentative curves of the intersecting strut $\{C^t_{Ek,l}\} = S_{Ek,l} \cap \Pi_k$. Trimming $\{C^t_{Ei,j}\}$ parts interior to $\Omega_{Ek,l}$ is then done by splitting curves at their intersection point with $\{C^t_{Ek,l}\}$, and discarding parts interior to $\Omega_{Ek,l}$. The test Midpoint $\left(C^t_{Ei,j,n}\right) \in \Omega_{Ek,l}$ is done to identify elements to discard.

Algorithm 3:

Generate slice curves of strut $E_{i,j}$ by trimming interior edges

Input: *plane* Π_k, strut $E_{i,j}$
Return value: $\{C_{Ei,j,n}\}$ = slice curves generated by strut $E_{i,j}$
$\{E_{k,l}\}$ = Get the list of struts that intersect strut $E_{i,j}$, such as $\Omega_{Ek,l} \cap \Omega_{Ei,j} \neq \emptyset$
$\{C^t_{Ei,j}\}$ = Get the tentative curves formed by the slice of strut's surface $S_{Ei,j} \cap \Pi_k$,
$\{C^t_{Ek,l}\}$ = Get the tentative curves formed by the slice of strut's surface $S_{Ek,l} \cap \Pi_k$.
$\left\{C^t_{Ei,j,n}\right\}$ = Split $\{C^t_{Ei,j}\}$ curves by inserting vertices at their intersection points with $\{C^t_{Ek,l}\}$.
$\{C_{Ei,j,n}\}$ = Discard elements of $\left\{C^t_{Ei,j,n}\right\}$ if they lie inside the interior of $\Omega_{Ek,l}$.

8 Slicing of caps spheres (step 4)

This process aims at the generation of arcs forming the slice curves lying on cap spheres S_{Vi}.

First, the set of tentative arcs $\left\{C^t_{Vi,j}\right\}$ is built by intersecting slice plane Π_k against each sphere S_{Vi} domain lying outside adjacent strut $\Omega_{Ei,j}$ (see Figs. 12a and 12b):

$$\left\{C^t_{Vi,j}\right\} = \left(S_{Vi}\Omega_{Ei,j}\right) \cap \Pi_k \forall j \in N(V_i).$$

The slice curves lying on sphere S_{Vi} are the common parts of tentative arcs in $\left\{C^t_{Vi,j}\right\}$ (see Figs. 12c and 13):

$$\{C^n_{Vi}\} = \cap_j C^t_{Vi,j} \forall j \in N(V_i).$$

Each tentative arc $\left\{C^t_{Vi,j}\right\}$ is constructed as following:

– Circle $C_i = S_{Vi} \cap \Pi_k$ is found.

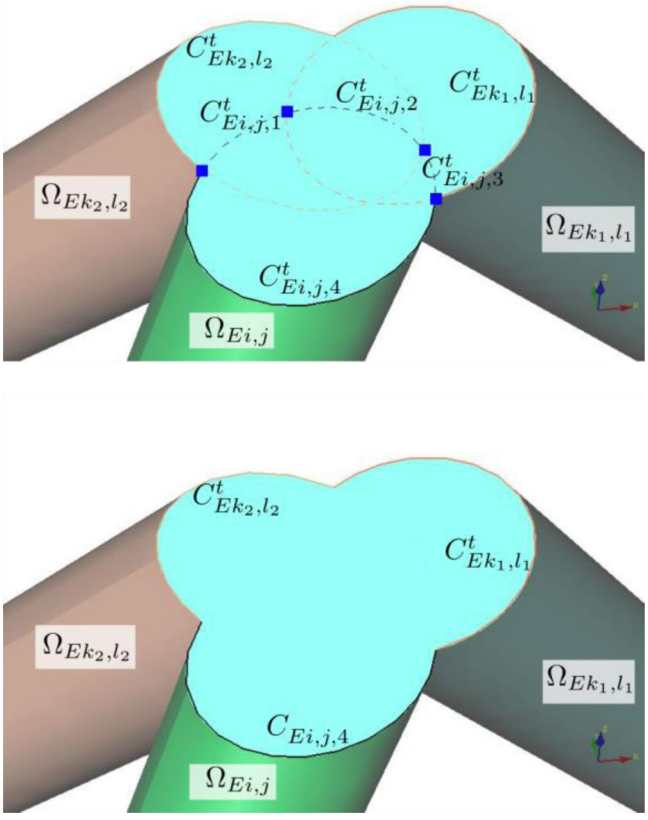

Fig. 11. Trimming interior edges.

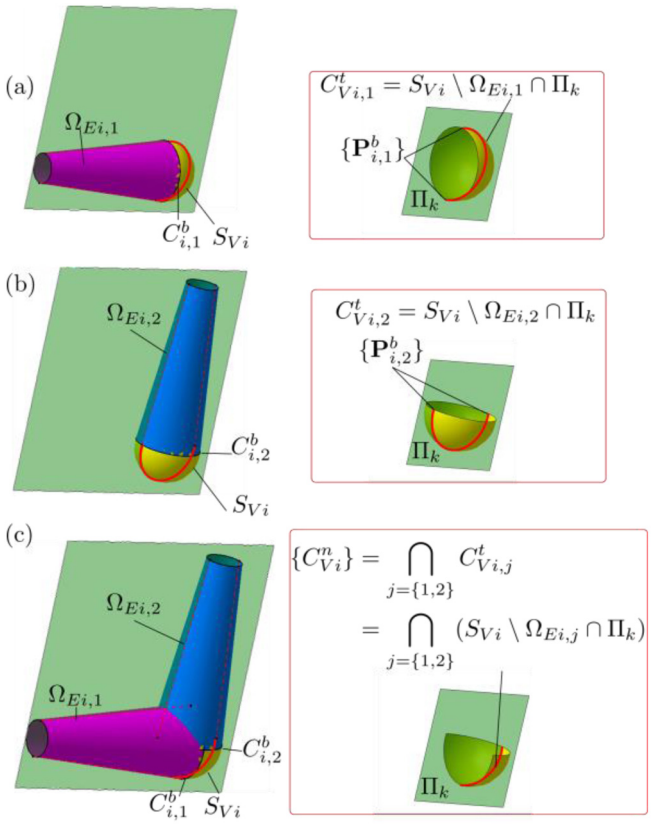

Fig. 12. slice curves lying on sphere S_{Vi} are the intersection of tentative arcs $\{C^n_{Vi}\} = \cap_j C^t_{Vi,j} \forall j \in N(V_i)$.

– End points $\partial C^t_{Vi,j}$ of tentative arc $C^t_{Vi,j}$ are located on circle $C^b_{i,j}$ which is the intersection of S_{Vi} and $S_{Ei,j}$ (see Figs. 12 and 13). As $C^t_{Vi,j}$ lies inside the slice plane Π_k (see Fig. 12):

$$\partial C^t_{Vi,j} = \left\{\mathbf{P^b_{i,j}}\right\} = \{C^b_{i,j}\} \cap \Pi_k.$$

Tentative arc $C^t_{Vi,j}$ is the part of C_i bounded by $\left\{\mathbf{P^b_{i,j}}\right\}$ and lying *outside* $\Omega_{Ei,j}$.

Tentative arcs $C^t_{Vi,j}$ belong to different parametric intervals of the same circle C_i, defined as $[L_{i,j}; U_{i,j}] \subset [-\pi; 3\pi]$ where $L_{i,j} \leq U_{i,j} + 2\pi$ (see Fig. 13).

Parametric intervals of slice curves are obtained by intersecting tentative arcs intervals all together:

$$\left\{\left[L^n_i; U^n_i\right]\right\} = \cap_j\left(\left[L_{i,j}; U_{i,j}\right]\right).$$

9 Examples and discussion

Test-case #1: this stem lattice structure illustrated in Figure 14 has conical struts with radii varying in [0.4 ; 1.2] mm. This test aims at demonstrating that the slices generated by our slicer are faithful, and fast generated. The lattice structure contains 4700 struts. The 324 layers have been generated in 795 ms. The layers have been saved in a CLI formatted file, and has been readed successfully using netfabb software. The computation time of traditional CAD+STL involves generating the B-Rep model of the truss structure, exporting it in STL format, and subsequently generating slices from the STL file. The time with traditional CAD+STL slicing approach is 96 hours.

The algorithm efficiency has been tested on octet truss lattice structures having the cell pattern repeated $a = 1$, 2, 4, 8 times along each axis (see Fig. 15).

The layer thickness was set to 25 µm, and the cell size was 10 mm (maximum distance along X,Y,Z axis between nodes of a unit-cell), strut diameter was 1 mm.

In Figure 16, the total time of skeleton lattice creation and direct slicing was compared with the traditional CAD +STL slicing approach. CAD files were built on CATIA V5 using the pattern copy feature in X, Y and Z directions. The B-Rep CAD model of the lattice was tessellated using a 0.01 mm sag error and exported in a STL file. Then, CuraEngine [19] processed STL files. The STL slicer generates rough polygons while direct skeleton slicing generates exact slicing curves.

The graph in Figure 17 shows that the trend curve of CAD+STL execution time grows as a^4 due to the CAD model generation time, while the direct slicing approach grows as a^3 which is much faster. As the number of intersected struts is proportional to a^2 and the number of slices is proportional to a^1, the overall slicing complexity is proportional to a^3. The direct slicing time verify this cubic

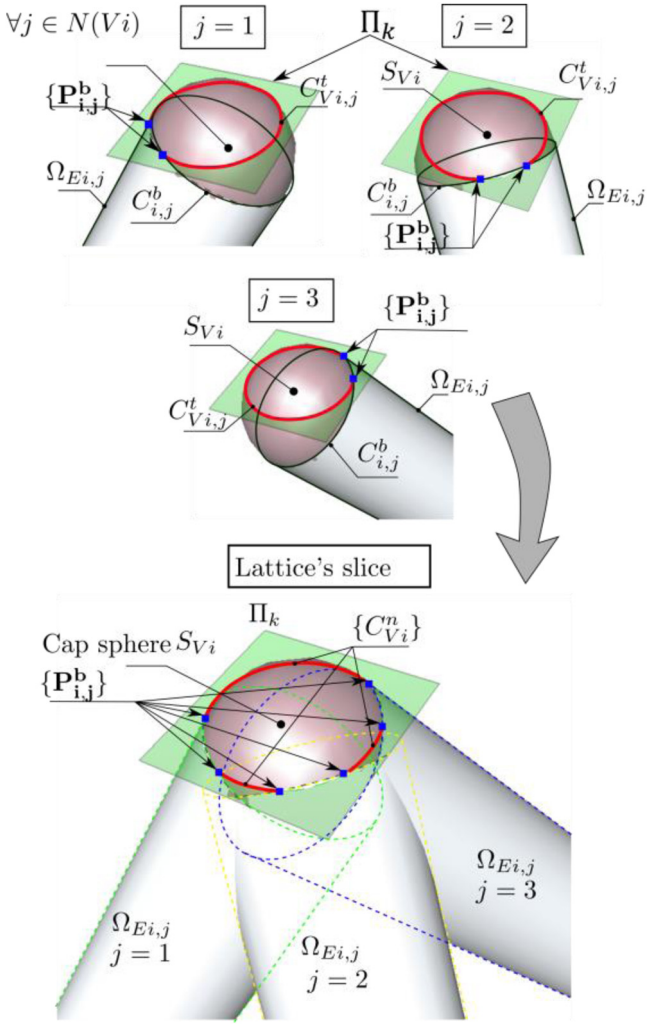

Fig. 13. Slice on a sphere.

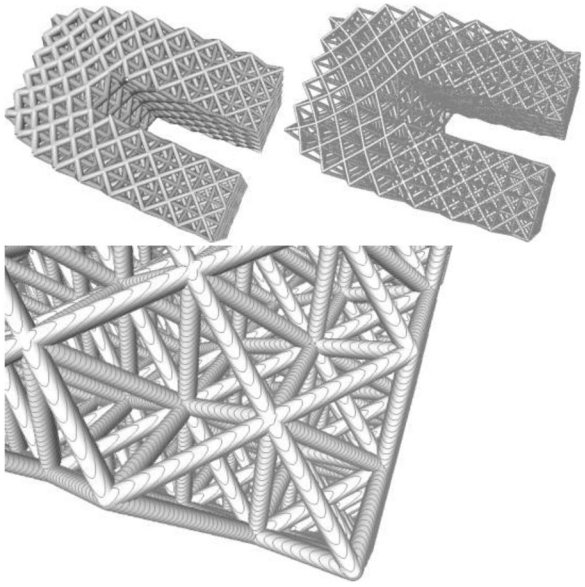

Fig. 14. Slices on the lattice structure of a stem.

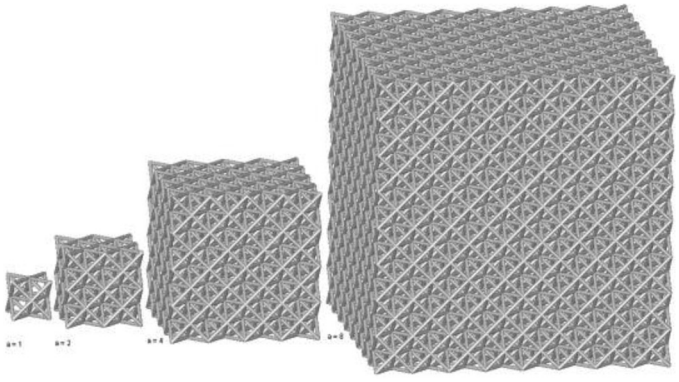

Fig. 15. Octet truss lattice structures for X,Y,Z pattern count a = 1, 2, 4, 8, each cell has 10mm side length, and 1mm of strut diameter.

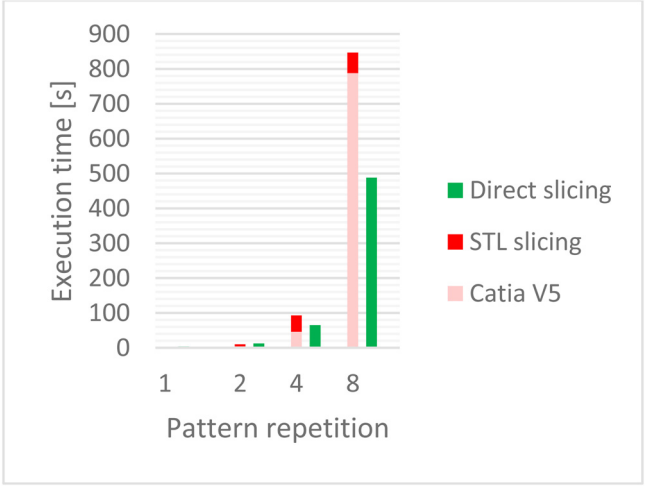

Fig. 16. Comparison of Catia V5 + STL slicing runtimes and between the proposed direct slicing method runtimes.

complexity behavior as the time scales cubically with a. In addition, the part-cutting process is simplified, from four steps with the traditional CAD + STL approach to just two with the proposed approach. This method eliminates the need to deal with intersections between lattice beams and spheres, as well as the STL file generation steps, as demonstrated in Figure 18.

10 Conclusion and perspectives

The direct slicing algorithm developed here outperforms the conventional method of producing and then slicing STL description of the part geometry, for sphere-cone-cylinder based lattice structures. The method is well adapted to lattice structures having struts with variable diameters, and can be extended to process lattice structures having different strut size incident to the same node.

A possible direct extension of this method is the creation of blends directly on slice curves located near struts joints, to avoid stress concentration compromising the strength of the structure.

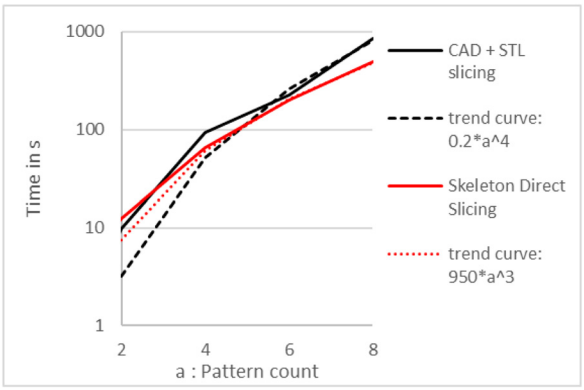

Fig. 17. Time comparison between skeleton slicing and CAD +STL slicing.

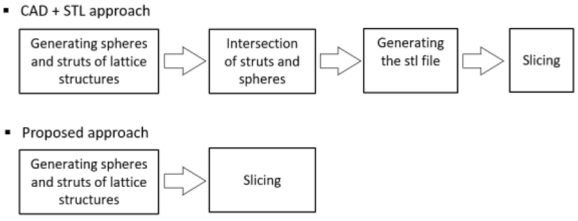

Fig. 18. Comparison of the CAD + STL approach with the proposed approach.

Another extension of this work would be the application of the proposed algorithm to Quador surfaces and Quador struts, which are currently not implemented in the OpenCascade CAD kernel [20].

Funding

The Article Processing Charges for this article are taken in charge by the French Association of Mechanics (AFM).

Conflicts of interest

The authors state that they don't have any conflicts of interest with respect to the research, authorship, or publication.

Data availability statement

All data generated or analyzed during this study are included in the present article.

Author contribution statement

Gilles Foucault: Methodology, Software development, Writing – original draft, Validation. Pierre-Thomas Doutre: Writing – review & editing, test lattice-structure "stem", Validation. Frédéric Vignat: review & editing, Methodology.

Appendix A

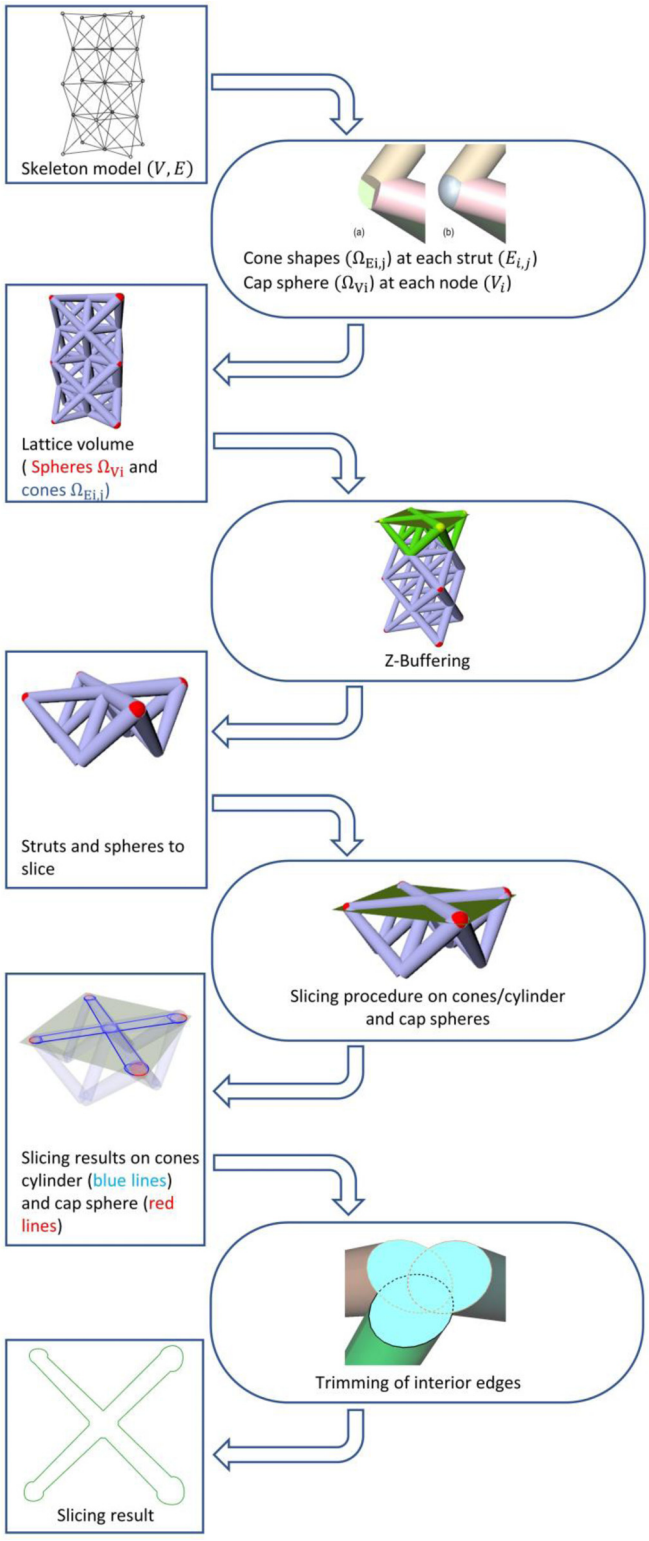

Fig. A.1. Graphical abstract.

References

[1] M. Suard et al., Mechanical equivalent diameter of single struts for the stiffness prediction of lattice structures produced by Electron Beam Melting, Addit. Manufactur. **8**, 124–131 (2015)

[2] A.H. Azman, F. Vignat, F. Villeneuve, CAD tools and file format performance evaluation in designing lattice structures for additive manufacturing, J Teknologi **80** (2018). doi: 10.11113/jt.v80.12058

[3] A.H. Azman, F. Vignat, F. Villeneuve, D.S. Nguyen, Creation of lattice structures with skeleton model for additive manufacturing, Int. J. Interactive Des. Manufactur. **15**, 381–396 (2021)

[4] M.K. Thompson et al., Design for additive manufacturing: trends, opportunities, considerations, and constraints, CIRP Ann. **65**, 737–760 (2016)

[5] J. Gardan, Additive manufacturing technologies: state of the art and trends, Int. J. Product. Res. **54**, 3118–3132 (2016)

[6] F. Bianconi, Bridging the gap between CAD and CAE using STL files, Int. J. CAD/CAM **2**, 55–67 (2002)

[7] Y. Chen, A mesh-based geometric modeling method for general structures, in *IDETC-CIE2006*, Volume **3**: 26th Computers and Information in Engineering Conference (2006) p. 269–281

[8] Slic3r – Open source 3D printing tool box. [En ligne]. Disponible sur: https://slic3r.org/

[9] W.E. Lorensen, H.E. Cline, Marching cubes: a high resolution 3D surface construction algorithm, in Proceedings of the 14th Annual Conference on Computer Graphics and Interactive , in SIGGRAPH '87. New York, NY, USA: ACM (1987), p. 163–169

[10] L. Chougrani, J.-P. Pernot, P. Véron, S. Abed, Lattice structure lightweight triangulation for additive manufacturing, Comput.-Aid. Des. **90**, 95–104 (2017)

[11] G. Savio, R. Meneghello, G. Concheri, Optimization of lattice structures for additive manufacturing technologies, in Proceedings of the International Joint Conference on Mechanics, Design Engineering & Advanced Manufacturing (JCM 2016), in Lecture Notes in Mechanical Engineering. Catania, Italy: Springer Verlag (2016)

[12] M.C. Messner, A fast, efficient direct slicing method for slender member structures, Addit. Manufactur. **18**, 213–220 (2017)

[13] C. Yong, Regulating complex geometries using layered depth-normal images for rapid prototyping and manufacturing, Rapid Prototyp. J. **19**, 253–268 (2013)

[14] B. Starly, A. Lau, W. Sun, W. Lau, T. Bradbury, Direct slicing of STEP based NURBS models for layered manufacturing, Comput. Aided Des. **37**, 387–397 (2005)

[15] A. Gupta, G. Allen, J. Rossignac, QUADOR: QUADric-Of-Revolution beams for lattices, Comput. Aided Des. **102**, 160–170 (2018)

[16] A. Gupta, G. Allen, J. Rossignac, Exact representations and geometric queries for lattice structures with quador beams, Comput. Aided Des. **115**, 64–77 (2019)

[17] S.S. Mustafa, I. Lazoglu, A new model and direct slicer for lattice structures, Struct. Multidisc. Optim. **63**, 2211–2230 (2021)

[18] J.A. Bondy, U.S.R. Murty, Graph theory with applications, Fifth printing, 1982. Elsevier Science Publishing Co., Inc. (1976)

[19] Ultimaker Cura: Powerful, easy-to-use 3D printing software, ultimaker.com. https://ultimaker.com/software/ultimaker-cura

[20] Open CASCADE Technology | Collaborative development portal. Consulté le: 19 avril 2024. https://dev.opencascade.org/

Cite this article as: G. Foucault, F. Vignat, P.-T. Doutre, A direct slicing method for optimized lattice structures, Mechanics & Industry **25**, 26 (2024)