Lattice structure lightweight triangulation for additive manufacturing

Laurent Chougrani[a,b], Jean-Philippe Pernot[a,*], Philippe Véron[a], Stéphane Abed[b]

[a] *Arts et Métiers ParisTech, LSIS laboratory UMR CNRS 7296, France*
[b] *Poly-Shape, France*

Abstract

Additive manufacturing offers new available categories of geometries to be built. Among those categories, one can find the well developing field of lattice structures. Attention has been paid on lattice structures for their lightweight and mechanical efficiency ratio, thus leading to more optimized mechanical parts for systems. However this lightness only holds true from a mass related point of view. The files sent to additive manufacturing machines are quite large and can go up to such sizes that machines can freeze and get into malfunction. This is directly related to the lattice structures tendency to be of a high geometric complexity. a large amount of vertices and triangles is necessary to describe them geometrically, thus leading to larger file sizes. With the increasing use of lattice structures, the need for their files to be lighter is also rising. This paper aims at proposing a method for tessellating a certain category of such structures, using topologic and geometric criteria to generate as few as possible triangles, thus leading to lightweight files. The triangulation technique is driven by a chordal error that control the deviation between the exact and tessellated structures. It uses interpolation, boolean as well as triangulation operators. The method is illustrated and discussed through examples from our prototype software.

Keywords: Lattice structures, Triangulation, 3D modeling, Lightweight STL files, Additive Manufacturing.

1. Introduction and related works

Additive manufacturing is a promising technology that allows to create more complex shapes than ever before. Among the new available convoluted features, one can find lattice structures that usually consist of an interconnected set of beams. Lattices are generally lightweight structures and can have very interesting mechanical properties [1]. As aforementioned, lattice structures are gaining more and more interest in the industry, and designers or engineers are now confronted with their manipulation, as they are considered as a revolution in conception paradigms [2, 3]. Furthermore, today's applications require lattice structures to contain more and more beams and have smaller and smaller features leading to several difficulties for 3D rendering and manipulating such complex structures. As a result the triangulation of these complex structures has become crucial for additive manufacturing as machines use tessellated files to print objects.

Triangulation is a well-studied field of 3D modeling as it is needed for finite elements methods or visualization purposes. Different triangulation techniques have been devised. When exact boundaries of the desired structure are available, e.g. a B-Rep CAD model, they can be used as constraints for the triangulation which can then be performed using the Constrained Delaunay Triangulation (CDT) [4, 5]. Such a method allows to get a low amount of triangles for a given structure, but they need the exact geometry and its boundaries as an input. Thus, computation times can be high especially for lattice structures, having numerous boundaries (Figure 1).

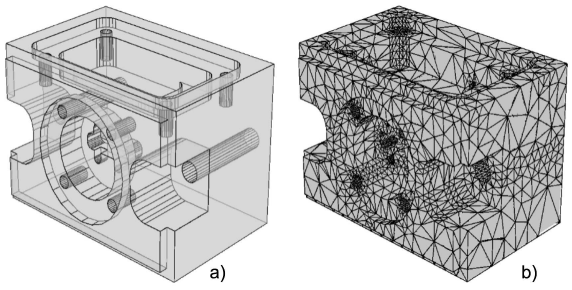

Figure 1: Illustration of CDT: (a) exact boundaries, (b) triangulation.

To circumvent some of those issues, Marching Cubes Method (MCM) tends to be more and more used to tessellate 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 field [6]. The field can be deduced from a skeleton (e.g. points, wireframe, surface, volume). Thus, there is no need to have an ex-

*Corresponding author
Email address: jean-philippe.pernot@ensam.eu (Jean-Philippe Pernot)

act boundary of the object. However, it usually generates a high amount of triangles which are usually of irregular geometric quality throughout 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. Adaptive voxel grids could lead to marching cubes that would be less dependent on the beams orientations [7]. However, the grid would need to be computed before the triangulation begins and would lead to some extra computation time. The quality can also be improved afterwards using some recent methods such as [8]. MCM has been widely used for medical imagery, as shapes of organic structures are often convoluted [9] and their exact boundaries rarely available (Figure 2).

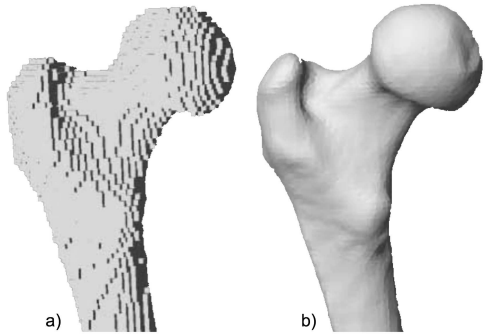

Figure 2: Illustration of MCM: (a) voxel grid, (b) triangulation [9].

Lattice structures can be defined as complex regarding the high number of vertices/triangles needed to define, render and manufacture them with an arbitrary precision. This huge amount of data can lead to several difficulties and notably: (i) software can have difficulties rendering a huge amount of triangles, (ii) additive manufacturing machines can have difficulties leading to malfunction when files get too heavy (ordinarily around 2Gb).

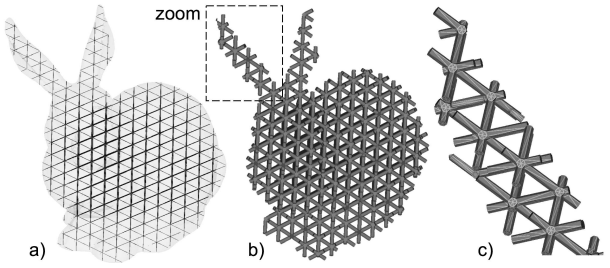

Figure 3: Illustration of LSLT: (a) 3D graph associated to a lattice structure inside the Stanford bunny, (b) lattice structure triangulation, (c) zoom on the ear.

This paper tackles the second difficulty and introduces a new approach for direct triangulation of lattices structures. Contrary to CDT, the algorithm does not need an exact representation of boundaries, but only the 3D graph that represents the lattice (figure 3). Three main operators have been defined to control efficiently the number of generated triangles: interpolation, boolean and triangulation operators. The contribution is threefold : (i) the algorithm generates much less triangles than CDT and MCM; (ii) it is not sensitive to the object's orientation with respect to the reference frame; (iii) quality and accuracy of the triangulation can be controlled separately. Section 2 introduces the problem, notations as well as the main ideas driving the different steps of the so-called Lattice Structure Lightweight Triangulation (LSLT) algorithm detailed in Section 3. The proposed approach is compared to CDT and MCM triangulation techniques and results are discussed using several academic and industrial examples introduced in Section 4. The last section refers to the different contributions as well as discussing future.

2. Problem formalization and notations

In the proposed approach, the idea is to triangulate lattice structures directly through their associated 3D graph using topologic and geometric informations. In order to reduce rendering difficulties, this triangulation can be performed at the very end of the design process, i.e. just before uploading the file to the machine.

This manuscript focuses on a certain type of lattice structures, which consist of a set of interconnected beams (i.e. unidimensional elements) that forms a 3D graph. this graph can be triangulated while considering each beam as a cylinder. The topology of the lattice is defined through the adjacency matrix associated to its 3D graph using graph theory [10]. For sake of clarity, as both graph and triangulation are composed of vertices, 3D graph's vertices are her referred to as nodes and denoted N, and triangulation's vertices are called vertices and denoted V. Additional subscripts will be used to further characterize them.

The 3D graph adjacency matrix is a $n \times n$ symmetric matrix $C = (c_{ij})$ with n the number of nodes in the lattice $\{N_1, \cdots, N_n\}$ and c_{ij} the number of beams between two nodes N_i and N_j. In particular, $c_{ij} = 0$ if no beams are connecting the two nodes N_i and N_j. Theoretically, the terms c_{ij} can be greater than 1 if there are some loops, i.e. a node connected to itself [11], but this will not happen in our case and the adjacency matrix will only be composed of 0 and 1. Thus, each beam B_{ij} connects exactly two nodes N_i and N_j (Figure 4). The direction of a beam is given by a unit vector $\boldsymbol{d}_{ij}$. Its length is denoted ℓ_{ij} and the angle between two beams B_{ki} and B_{ij} is denoted α_{kij} where i is the index of the central node. A unique radius R_i is associated to each node N_i and defines the common radius of all the sections connected to it. Sections are defined per beam, with their first index indicating the node from where it is considered. Thus, σ_{ij} refers to the extremity section of B_{ij} connected to N_i. Finally, the number of beams incom-

ing to a node N_i is the node valence denoted Val_i. On the example of Figure 4, $Val_i = 3$, $Val_j = 4$ and $Val_k = 1$.

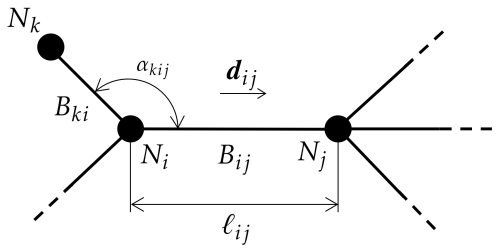

Figure 4: Simple lattice structure and associated notations.

To control better the quality of the triangulation between two extremity sections of a beam B_{ij}, two control parameters are introduced (Figure 5):

- the number n_{ij} of intermediary sections σ_{ikj}, with $k \in \{1, \cdots, n_{ij}\}$, which can be optionally inserted between N_i and N_j. Those additional sections have radii denoted r_{ikj}.
- the number n_{ikj} of vertices $\boldsymbol{V}_{ikjs}$, with $s \in \{1, \cdots, n_{ikj}\}$, in the intermediary section σ_{ikj}. Those vertices form an ordered list organized by angular positions, with an identical zero angular position for all the sections of a beam.

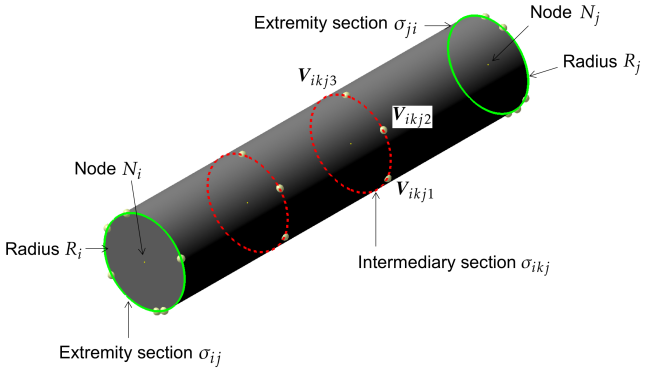

Figure 5: Beam B_{ij} represented as a cylinder with a constant radius for all the circular sections.

Those definitions combined with the positions of the nodes in the 3D space are sufficient to have a fully defined lattice structure.

Considering that beams are modeled as cylinders connected to extremity nodes, trying to triangulate them as a simple cylinders would lead to difficulties in two different ways that can be seen on Figure 6: (i) overlapping triangles leading to a non-manifold mesh; (ii) holes/gaps in the mesh. We want to draw attention to the fact that for rendering purposes, cylinders could be triangulated separately with no regard to interference. But for printing applications this would lead to the two following main problems:

- The slicing operation needed to print the part and consisting in making cross sections of the part, would probably fail, due to the overlap, the slicing software would have a hard time to sort out what is inside and outside the part.
- Even if the slice operation is a success, the overlapping area would lead to the machine "Laser beaming" several time the same surface, causing over-fused "blobs" around the lattice node and leading to degraded material properties.

The problem of overlapping triangles is solved either by CDT and MCM. However, the gap issue is not tackled by CDT, but is resolved by MCM as it is based on an Euclidean distance field which rounds all sharp edges. Thus, the need for a method that can both generates lightweight and manifold meshes with no holes or gaps leads to the description of our Lattice Structure Lightweight Triangulation (LSLT) method.

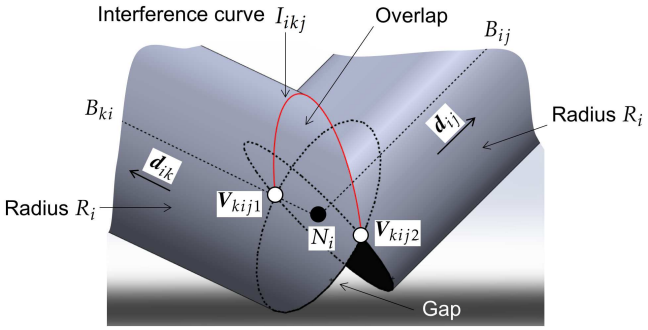

Figure 6: Interference around the extremities of two connected beams B_{ij} and B_{ik}.

3. Lattice Structure Lightweight Triangulation (LSLT)

The proposed LSLT technique consists in six main steps illustrated on the flowchart of Figure 7 and detailed in the next subsections: (i) creation of the mandatory triangulation vertices from an input lattice structure (section 3.1); (ii) use of a chordal error criterion to interpolate new vertices (section 3.2); (iii) displacement of the vertices using a special boolean operator (section 3.3); (iv) optionally interpolate the extremity sections to create additional sections and vertices (section 3.4); (v) triangulation of all the sections (section 3.5); (vi) filling of the holes (section 3.6).

3.1. Mandatory vertices search

First step is to define the location of triangulation vertices on the extremity sections. It is done by parsing through the nodes and sorting out every duet of beams $\{B_{ij}; B_{ik}\}$ around each given node N_i. For a duet of beams, one can see that the two ideal cylinders will have interferences. Those interferences occur for all connected beams

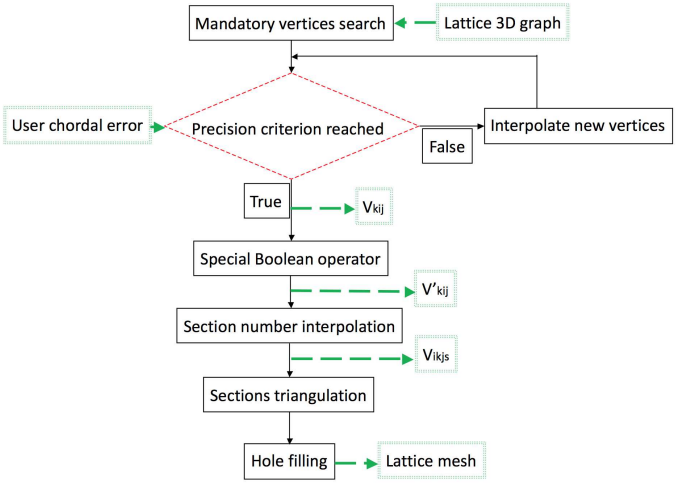

Figure 7: Overall framework of the LSLT technique

at a given extremity as shown in Figure 6. The two given interfering cylinders possess their original extremity sections σ_{ij} and σ_{ik} as well as an interference curve I_{ikj} which is normally obtained using boolean operators.

To get a manifold triangle mesh, the triangulation vertices must lay on this interference curve. In case the two beams are not collinear, only two points $\boldsymbol{V}_{kij1}$ and $\boldsymbol{V}_{kij2}$ are already lying on I_{kij} and are defined as follows:

$$\begin{cases} \boldsymbol{V}_{kij1} = \boldsymbol{N}_i + R_i.\boldsymbol{O}_{kij} \\ \boldsymbol{V}_{kij2} = \boldsymbol{N}_i - R_i.\boldsymbol{O}_{kij} \end{cases} \text{ with } \boldsymbol{O}_{kij} = \frac{\boldsymbol{d}_{ik} \wedge \boldsymbol{d}_{ij}}{\|\boldsymbol{d}_{ik} \wedge \boldsymbol{d}_{ij}\|} \tag{1}$$

Here : "$\wedge$" is the symbol used for cross product. One can then define these two mandatory triangulation vertices for the two extremity sections that lay on N_i. Looping through the duets of beams will give all mandatory triangulation vertices. Each duet gives two points, so for a given node N_i, the number of mandatory triangulation vertices is at most:

$$2 \times \frac{Val_i!}{2!(Val_i - 2)!} \tag{2}$$

Due to symmetry conditions some mandatory triangulation vertices can be redundant and will only be taken into account once. These vertices being collected and stored in each corresponding sections, one will understand that they all lay on the sphere around N_i of radius R_i. Due to the angular incidence of each beam, the sphere represents the minimum surface on which the real triangulation vertices must lay. The proof of that statement can be made following some elegant ideas found in [12].

Let $\boldsymbol{d}_{ij}$ and $\boldsymbol{d}_{ik}$ be the unit vectors of the two cylinders, forming an angle α_{kij} that can be limited to $[0,\pi/2]$ due to symmetry, and R_i their common radius as radii are defined at each node. As we have only two cylinders, one can simplify the problem by calling xOz the plane that contains them both. The problem is to solve the intersection as a $f(z)$ function. Using those simplifications, one can write the following cylinder parametric equation (first for two perpendicular cylinders):

$$\begin{cases} y^2 + z^2 = R_i^2 \\ x^2 + y^2 = R_i^2 \end{cases} \tag{3}$$

This equation can then be transformed using $R(v, \alpha_{kij})$ the rotation operator around the positive y axis of value α_{kij}, and using the linear transformation to get the system which solution is the intersection curve for any arbitrary value of α_{kij}.The idea is to fix one cylinder and let the other free of rotation in order to get the intersection curve for any values of α_{kij}.

$$R\left(x, y, z, \alpha_{kij}\right) = \begin{bmatrix} \cos(\alpha_{kij}) & 0 & -\sin(\alpha_{kij}) \\ 0 & 1 & 0 \\ \sin(\alpha_{kij}) & 0 & \cos(\alpha_{kij}) \end{bmatrix} \begin{bmatrix} x \\ y \\ z \end{bmatrix} \tag{4}$$

$$\begin{cases} y^2 + (x.sin(\alpha_{kij}) + z.\cos(\alpha_{kij})^2 = R_i^2 \\ \\ x^2 + y^2 = R_i^2 \end{cases} \tag{5}$$

After eliminating y^2 and doing some arrangements, one can get a nice parametric equation:

$$\begin{cases} x = R_i.\cos(t) \\ y = R_i.\sin(t) \\ z = R_i.\sin(t)[1 + \sec(\alpha_{kij})] \end{cases} \quad \forall t \in [-\pi, \pi] \tag{6}$$

From that system, one can write the minimum distance to the extremity node (here $(0,0,0)$) as:

$$d = \sqrt{x^2 + y^2 + z^2} \tag{7}$$

With respect to the values found for (x,y,z), one get the distance expressed as :

$$d(t) = R_i.\sqrt{1 + \sin(t)^2(1 + \sec(\alpha_{kij}))} \ \ \forall t \in [-\pi, \pi] \tag{8}$$

Studying this distance is made easy as we have practical constraints on $\alpha_{kij} \in [0, \pi/2]$ and $t \in [-\pi, \pi]$. It comes:

$$\min d(t) = R_i \tag{9}$$

which means that the sphere of radius R_i is the minimum surface where mandatory triangulation vertices can lay for any couple of beam around a given node N_i.

One can also observe that this parametric equation would need to be performed for all couples of beams, and then having all those systems merged to get the whole set of interference curves. This is what is needed for constrained Delaunay triangulation and can be time consuming. The proposed method only needs all the $\boldsymbol{V}_{kij1}$ and $\boldsymbol{V}_{kij2}$ madantory vertices at each node N_i. One thus know that those points would only need to be pushed outward the sphere and never pulled inward. Those vertices need to be pushed depending on the value of α_{kij} as it will be explained in section 3.3.

3.2. Chordal error to interpolate new vertices

Once mandatory triangulation vertices have all been placed, a chordal error can be computed comparatively to the perfect circle of a cylindrical section. Depending on the value, extra vertices can be inserted to meet a certain criterion on shape approximation. They are added on the common sphere, and distributed upon all sections. This issue along with the number of section determination will be developed in section 4 as an analysis and comparison between LSLT and MCM. It is now needed to push those vertices onto the intersection curves. To do so, a simplification can be made, by noticing that for a duet of beams, the intersection curve always lays on a plane. This result could also be extracted from the system above.

3.3. Special boolean operator

In this section, to better illustrate the process and the way the operator acts on section's vertices, figures will use triangulated cylinders even though the triangulation is performed later in the process (section 3.4).

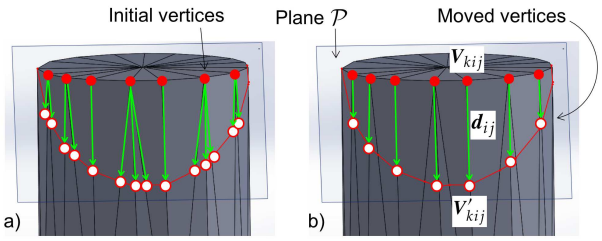

Figure 8: Boolean operation (subtraction) between a cylinder mesh and a plane: (a) classic boolean result with deletion and reconstruction of vertices along edges, (b) LSLT result with displacement of vertices along the unit vector $\boldsymbol{d}_{ij}$

.

Now that sections are completed and the chordal error criterion is met, one needs to get the section vertices on the interference curves. To do so a special boolean operator has been developed and uses the 3D graph to move some vertices around, thus simulating shape deformation. This operator works on each beam and consists in pushing vertices onto a plane along the beam axis unit vector $\boldsymbol{d}_{ij}$. The plane $\mathcal{P}_{kij}$ is defined between two connected beams B_{ij} and B_{ki}. Then, a point in $\mathcal{P}$ is referred as $\boldsymbol{V}_{\mathcal{P}}$.

To understand its importance, a comparison with a classic boolean operator is proposed. As a simple example, let's consider a mesh intersected by a plane $\mathcal{P}$ (Figure 8.a). Classic boolean operators would: (i) look for intersection between the mesh elements and the plane; (ii) suppress triangles in these areas; (iii) use different methods to re-mesh such as subdivision-based or voxellization-based remeshing [13, 14, 15]. The resulting mesh would then be manifold but have a larger amount of triangles (Figure 8.a). It is here proposed to create a boolean operator that does not add triangles to the mesh, and only affects the position of vertices, hence preserving triangle connections and neighborhood data. Knowing that the geometry is composed of cylinders, the idea is to move each vertex of a section along the axis of the cylinder defined by the unit vector $\mathbf{d}_{ij}$. As a result, the number of triangles is preserved (Figure 8.b). For the sake of clarity, Figure 8 illustrates this for the simple case of one beam intersected by one plane.

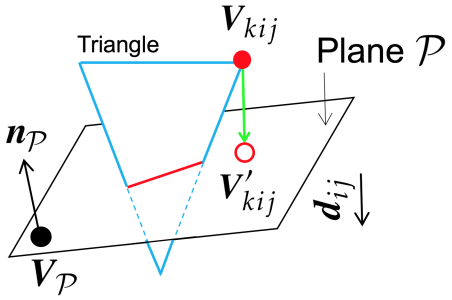

Figure 9: Vertices of extremity sections are pushed toward the plane.

Using the remark that each intersection curve always lay on a plane, it is proposed to use this plane as a cutting plane and push the section vertices upon it (Figure 9). Looping through the section's vertices, one can easily check if the considerate vertex is on the interference side or not. A vertex $\boldsymbol{V}_{kij}$ is moved along the vector $\boldsymbol{d}_{ij}$ of its beam if $[\boldsymbol{n}_{\mathcal{P}}.(\boldsymbol{V}_{kij}\text{-}\boldsymbol{V}_{\mathcal{P}})] < 0$, with $\boldsymbol{n}_{\mathcal{P}}$ the cutting plane normal unit vector such that $\boldsymbol{n}_{\mathcal{P}}.\boldsymbol{d}_{ij} > 0$, and $\boldsymbol{V}_{\mathcal{P}}$ a 3D point in the plane $\mathcal{P}$. Using radii defined per beam has for consequence that $\mathcal{P}$ is always the median plane. Note that if $\boldsymbol{V}_{kij}$ is on the other side of the plane it is not moved. One can compute the position $\boldsymbol{V}'_{kij}$ of the projection of $\boldsymbol{V}_{\boldsymbol{kij}}$ on the plane $\mathcal{P}$ (Figure 9):

$$\boldsymbol{V}'_{kij} = \boldsymbol{V}_{kij} + \lambda.\boldsymbol{d}_{ij} \quad \text{with} \quad \lambda = \frac{-\boldsymbol{n}_{\mathcal{P}}[\boldsymbol{V}_{kij} - \boldsymbol{V}_{\mathcal{P}}]}{\boldsymbol{n}_{\mathcal{P}}.\boldsymbol{d}_{ij}} \tag{10}$$

At this point, the different possibilities can be separated into three classes for a given node N_i:

- $Val_i = 1$. Nothing needs to be done as the beam is simply hanging.
- $Val_i = 2$. Section vertices are pushed at most one time toward a plane (as there is at most one intersection curve). The connection between the different section vertices is conserved. Performing a classic boolean successively on each section would lose the connectivity through adding new vertices (Figure 10).
- $Val_i > 2$. Section vertices might be pushed more than once. In this case holes might appear at the beam connection, as seen in Figure 11.c. One can also see that classic boolean would not form holes, and would lead to non manifold triangulation as some

vertices would lay onto triangle edges. The holes appearance is due to multiple cutting planes and can be understood through Figure 12. A solution is to add mandatory vertices to the section on the planes, but this would add more triangles than simply close the holes as discussed in section 3.6. Again, this operator works on vertices, the triangles are here displayed only for sake of clarity.

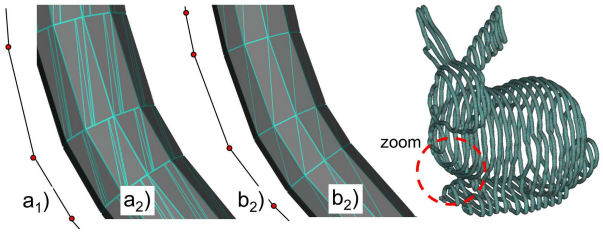

Figure 10: Comparison between: (a1) the lattice structure made of nodes and beams and (a2) its associated classic boolean result (non manifold and addition of triangles), (b1) the lattice structure and (b2) our associated special boolean operator result (manifold and no added triangles) for $Val_i = 2$ on the Stanford bunny lattice mesh.

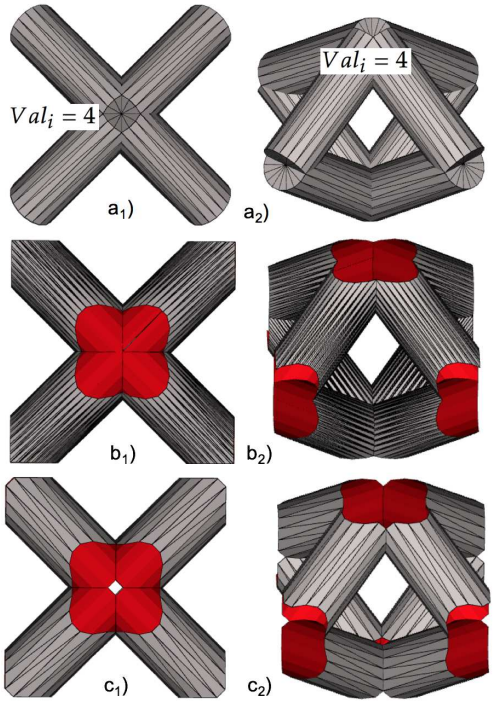

Figure 11: Top and isometric views for $Val_i = 4$ on a lattice cell mesh. Results comparison between : (a1) and (a2) initial cylinders with a given precision, (b1) and (b2) classic boolean result (non manifold and added triangles), and (c1) and (c2) our special boolean operator result (manifold and no added triangles)

3.4. Section number interpolation

Once all extremity sections are set up and the associated vertices moved on the different planes, intermediary sections can be added between the two extremity sections

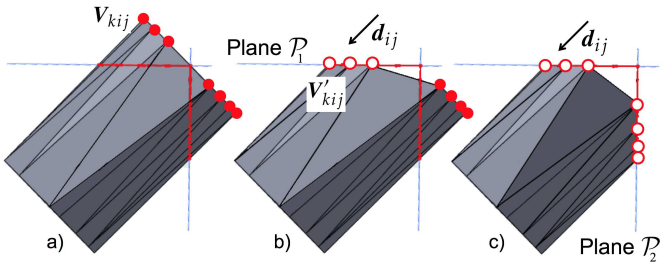

Figure 12: Different steps of our boolean operator : (a) initial section with intersection planes, (b) first displacements of vertices toward the first plane (gap creation), (c) second displacements toward the second plane with hole appearance.

of a beam B_{ij}. This objective is twofolds: (i) to generate a triangulation with a better and more homogeneous quality, in the sense that triangles will have their angles closer to 60°; (ii) to smoothen the possible angular distortion between two extremity sections of a cylinder. As Val_i can be different from Val_j, and α_{kij} different from α_{kji}, the two extremity sections may have both different number of mandatory vertices and different angular repartitions of these vertices.

As introduced in section 2, for a beam B_{ij}, two control parameters are used:

- the number n_{ij} of intermediary sections σ_{ikj} of radius r_{ikj}, with $k \in \{1, \cdots, n_{ij}\}$, located between the extremity nodes N_i and N_j.
- the number n_{ikj} of vertices $\boldsymbol{V}_{ikjs}$, with $s \in \{1, \cdots, n_{ikj}\}$, in the intermediary section σ_{ikj}.

First, the number n_{ij} of intermediary sections can be determined as follows:

$$n_{ij} = \mathrm{E}\left[\frac{\ell_{ij}}{\ell_{eij}}\right] \tag{11}$$

Where E is the function that returns the integer portion, ℓ_{ij} the length of the considered beam B_{ij} and ℓ_{eij} the mean length of an edge of the sections which can be approximated as follows:

$$\ell_{eij} \approx 2 \times \overline{R}_{ij} \times \sin\left(\frac{2\pi}{\overline{n}_{ij}}\right) \tag{12}$$

Where $\overline{n}_{ij}$ and $\overline{R}_{ij}$ are respectively the average of the number of vertices and the average of the radius of the two extremity sections.

Second, the number n_{ikj} of vertices in the intermediary section σ_{ikj} linearly evolves between the number of vertices of the two extremity sections. Thus, it results that:

$$n_{ikj} = \mathrm{E}\left[n_i + \frac{\ell_{ikj}}{\ell_{ij}}(n_j - n_i)\right] \tag{13}$$

With n_i (resp. n_j) the number of vertices in the extremity section σ_{ij} (resp. σ_{ji}) and ℓ_{ikj} the distance between the extremity section σ_{ij} and the considered intermediary section σ_{ikj}.

Finally, the vertices $\boldsymbol{V}_{ikjs}$ of the intermediary section σ_{ikj} are added in order to linearly smooth the angular deviation between the two extremity sections. If the numbers of vertices in the three sections are equal, i.e. $n_{ikj} = n_i = n_j$, then the following formula can be applied directly:

$$\boldsymbol{V}_{ikjs} = \boldsymbol{C}_{ikj} + r_{ikj} \cdot \frac{\boldsymbol{r}_{ikjs}}{\|\boldsymbol{r}_{ikjs}\|} \quad \forall s \in \{1, \cdots, n_{ikj}\} \tag{14}$$

with $\boldsymbol{C}_{ikj}$ the center of σ_{ikj}, and $\boldsymbol{r}_{ikjs}$ a vector obtained by linear combination so that:

$$\boldsymbol{r}_{ikjs} = \left(1 - \frac{k}{n_{ikj}}\right).\boldsymbol{r}_{ijs} + \frac{(k-1)}{n_{ikj}}.\boldsymbol{r}_{jis} \tag{15}$$

where $\boldsymbol{r}_{ijs}$ (resp. $\boldsymbol{r}_{jis}$) is a vector that goes from the center of section σ_{ij} (resp. σ_{ji}) to the vertex $\boldsymbol{V}_{ijs}$ (resp. $\boldsymbol{V}_{jis}$) of that section. If the number of vertices contained in each three sections are not equal, then the vectors $\boldsymbol{r}_{ijs}$ and $\boldsymbol{r}_{jis}$ are either averaged or extrapolated.

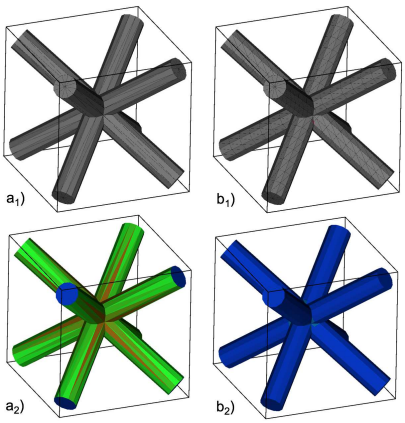

Figure 13: Influence of the number of intermediary sections on the quality of the triangulation : (a1) no intermediary sections and (a2) associated quality map, (b1) 8 intermediary sections and (b2) associated quality map.

Figure 13 shows the influence of intermediary sections on the quality of the resulting triangulation. Here, the quality of a triangle is defined as the ratio between its inner and outer circles [16]. The quality aspects are further discussed in the results section. Finally, one can say that even if the proposed approach allows for the treatment of multiple radii and number of vertices, the way the lattice structure is generated through the repetition of elementary cells often induces that the numbers of vertices in the different sections are equal and that the radii are also equal.

3.5. Sections triangulation

Once all extremities and intermediary sections are set, triangulation can be performed in two different ways depending on whether sections do or do not have the same number of vertices:

- If they have the same number of vertices, tessellation is performed directly, as vertices are angularly oriented, triangles will be added up two by two as part of quadrangles. Sections will also be tessellated two by two.
- If they don't have the same number of vertices, there is no direct way to sort out the best triangulation and a trick can be used. Sections are gathered together two by two. The second section is projected onto the plane of the first one with a positive homothetic scaling so that the two sections will not lay on the same circle (Figure 14). Then, in this plane, a Delaunay tessellation is performed between the two sections. This is done with an advancing front method from the first extremity to the second one, and then through re-deploying the sections along the axis, the 3D triangulation of the cylinder is obtained.

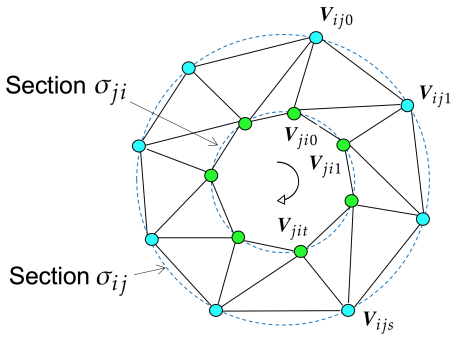

Figure 14: Two projected sections σ_{ij} and σ_{ji} with their unequal number of vertices $\boldsymbol{V}_{ijs}$ and $\boldsymbol{V}_{jit}$ and the resulting Delaunay triangulation.

3.6. Hole filling

Once every cylinder has been triangulated, and since cylinders are hollowed, holes appear in the mesh at extremity sections (Figure 11). Depending on the valance of a node N_i, several techniques can be used to fill in these holes:

- When $Val_i = 1$, the section is convex and one can simply chose arbitrarily a vertex in the section and connect it to all the other vertices, leading to a minimum number of triangles for the given section. Here, all the nodes of the section could also be directly connected to N_i.
- When $Val_i > 1$, several holes can appear and may not be convex depending on α_{kij}. Several techniques exist to fill in holes according to a surrounding mesh [17, 18, 19]. Here, a more simple technique is used. To fill in a hole h around a node N_i, the barycenter $\boldsymbol{G}_{ih}$ of the vertices forming the hole contour is first computed and pushed against the sphere of radius R_i centered in $\boldsymbol{N}_i$:

$$\boldsymbol{G}'_{ih} = \boldsymbol{N}_i + R_i \times \frac{\boldsymbol{G}_{ih} - \boldsymbol{N}_i}{\|\boldsymbol{G}_{ih} - \boldsymbol{N}_i\|} \tag{16}$$

The hole is then triangulated while connecting vertices of the contour to the projected barycenter $\boldsymbol{G}'_{ih}$.

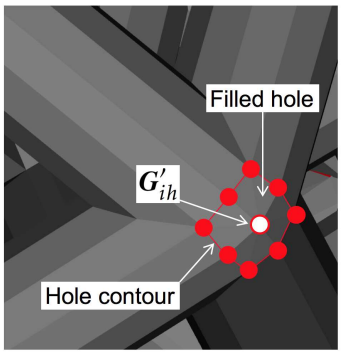

Figure 15: Hole filling using barycenter projection for $Val_i = 4$.

4. Results and discussion

LSLT technique has been applied to two academic (Stanford bunny and Stanford dragon) and one industrial (T-fork) models and compared with MCM and CDT. CATIA V5 has been used to compute the CDT and NTopology for the MCM. Section 4.1 discusses the results obtained when comparing the number of triangles generated by the three techniques for a given accuracy. Section 4.2 discusses the results obtained when trying to put as many beams as possible in a file of size 2Gb. Section 4.3 discusses the results in terms of triangles shape quality.

4.1. Number of generated triangles

Lattice structures to be triangulated are composed of cells repeated throughout space. The cells have the same topology as the one of Figure 11 for the three examples. However, the cells do not have the same size for the three examples (first row of Table 1). This is to take into account the size of the models themselves. The way these cells are generated is not detailed in this paper which focuses on the triangulation of the resulting lattice structures. Similarly, the radii of the beams as well as the approximation error (chord error) are not equal for the three models (rows 3 to 5 of Table 1).

	Bunny	T-fork	Dragon
Cell size (mm^3)	7×7×7	7×7×7	2×2×2
Beam radius (mm)	0.5	0.5	0.2
Chord error (% of radius)	5	5	5
Chord error (μm)	25	25	10

Table 1: Parameters used to compare the number of triangles generated by the algorithms on the three models.

Using Table 1 parameters, all three lattice structures can be triangulated using MCM, CDT and LSLT techniques. The triangulations are shown in Figures 16, 17 and 18 and the results are made available in Table 2. First, it can be noticed that the size of the file is directly proportional to the number of triangles. This is because of the adopted STL file format which stores the triangles one by one. Then, for a given example and chord error, results show that our LSLT technique generates much less triangles than the CDT and MCM. This was the first reason for developing this new triangulation technique.

		Bunny	T-fork	Dragon
Beams		3648	4605	12416
Triangles	MCM	924644	1130400	7462036
	CDT	377106	229888	Crashed
	LSLT	94872	126488	315728
File size (Mb)	MCM	44	53.9	355
	CDT	18.5	11.2	N/A
	LSLT	4.52	6.3	15
Gain LSLT wrt MCM		89.7%	88.8%	95.7%
Gain LSLT wrt CDT		74.8%	44.9%	N/A

Table 2: Gains in terms of triangles (or file sizes) when comparing our LSLT technique to MCM and CDT.

It also appears that some beams of the T-fork could not be computed using CDT. This is visible on Figure 17.c and it explains why the file is lighter for the T-fork than for the Bunny even-though the number of beams is greater for the T-fork than for the Bunny. This also explains the apparent gain diminution when comparing LSLT and CDT on the Bunny and T-fork. The results also shows that CDT has not been able to triangulate the lattice structure inserted into the Dragon. Throught the word "Crash" we simply state here the fact that after around 2 hours of computing, the software was still sorting the boundaries out and sending no information to windows. Windows detected it as a malfunction and shut down the software.. The MCM succeeds in triangulating the lattice structure of the Dragon but it generates much more triangles than LSLT does. Thus, the gain has risen up to 95.7% when comparing our approach to MCM. Marching cubes are sensitive to the ratio between the geometry dimension (both global and local) and the voxel grid size. Thus, as the density of the lattice increases, the refinement of the voxel grid increases to maintain the chord error, which results in a larger overall number of triangles and file size.

From those results, it becomes clear that CDT is not adapted in this case. To further compare the number of triangles generated by the LSLT and MCM techniques, the influence of the control parameters of each method can be analyzed. With LSLT, each time that an intermediary section is added up (section 3.4), a ring of triangles is attached to it. So, if $U_{LSLT}(0)$ represents the number of triangles when 0 intermediary section has been added, then it is obvious that the sequence is arithmetic and that its common difference is the number of triangles q per ring. Thus, if n intermediary sections are added, the number of triangles is directly:

$$U_{LSLT}(n) = U_{LSLT}(0) + n \times q \tag{17}$$

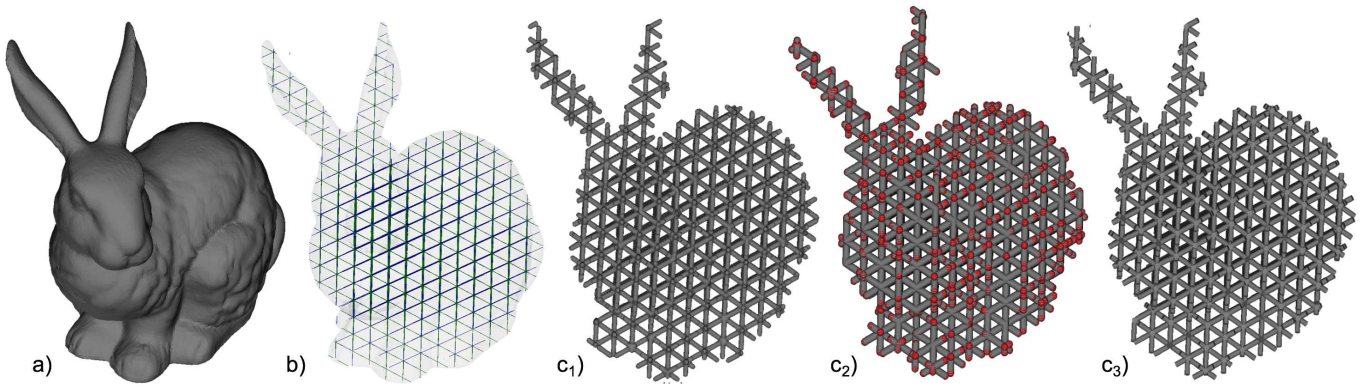

Figure 16: Triangulation of a lattice structure inserted in the Stanford bunny: (a) initial triangle mesh to be filled in by a lattice structure, (b) lattice cells of size $7 \times 7 \times 7$, (c_1) MCM triangulation: 924644 triangles, (c_2) CDT triangulation: 377106 triangles, (c_3) LSLT triangulation: 94872 triangles.

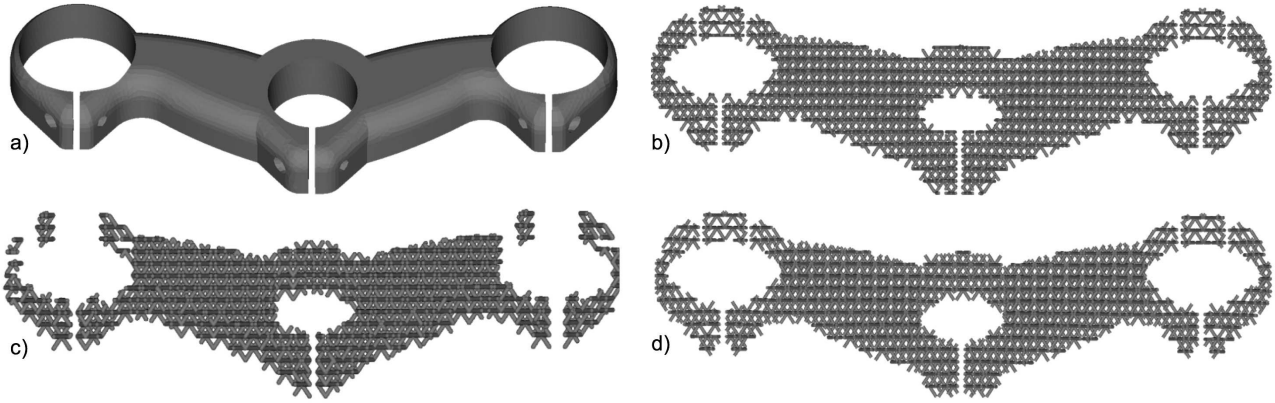

Figure 17: Triangulation of a lattice structure inserted in an industrial T-fork: (a) triangle mesh used to generate the lattice cells of size$7 \times 7 \times 7$, (b) MCM triangulation: 1130400 triangles, (c) CDT triangulation: 229888 triangles, (d) LSLT triangulation: 126488 triangles.

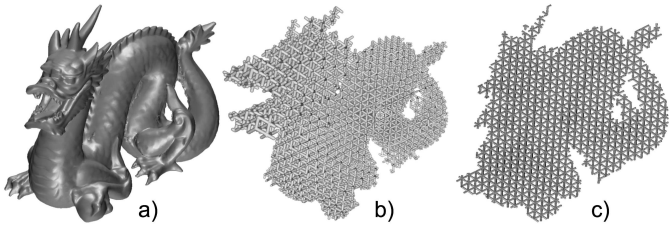

Figure 18: Triangulation of a lattice structure inserted in the Stanford dragon: (a) initial triangle mesh used to generate the lattice cells of size $2 \times 2 \times 2$, (b) MCM triangulation: 7462036 triangles, (c) LSLT triangulation: 315728 triangles.

Using the same idea, one can understand when the number of vertices per section increases, the total number of triangles raises linearly. Thus, each time a vertex is added to the sections the number of triangles evolves such as:

$$U_{LSLT}(n) = U_{LSLT}(0) + n \times [2 \times (s-1)] \tag{18}$$

With s the number of sections. The common difference is due to the fact that adding one vertex to each section leads to the creation of two triangles between two consecutive sections.

With MCM, identifying the evolution of the number of triangles is trickier as there are different cases and different ways of refining the grid. Here, it is assumed that the grid is refined by linear subdivision, and that the beam orientation is constant throughout the lattice. For a cylindrical iso-surface, which is the case when modeling the beams of our lattice structures, and a grid constructed such as the cube length is smaller than the beam radius, then each cube is at most crossed once by the iso-value. From one subdivision to another, each cube is subdivided into 8 sub-cubes, each sub-cube is then crossed at most once by the iso-surface. This means that the number of triangles rises from one to q, with $1 \leq q \leq 8$. Actually, the value of q depends on the shape of the iso-surface, its orientation relatively to the voxel grid and on the voxel grid starting position. The value of q varies through iterations, nevertheless the number of triangles at a given iteration is obtained through a multiplication of this number at the previous iteration. Therefore, the sequence is geometric this time and can be represented such as:

$$U_{MCM}(n) = U_{MCM}(0) \times q^n \tag{19}$$

As a conclusion, LSLT generates much less triangles for a given chord error than the two other methods. Fur-

thermore, when reducing the chord error, the number of triangles generated by MCM increases much faster than LSLT. Thus, the gain of LSLT with respect to MCM increases as the chord error decreases. Using LSLT one can first set the section vertices in order to meet the chord error criterion, and then directly compute the right number of intermediary sections to ensure a certain quality of the triangles. The triangles shape quality is discussed in section 4.3.

4.2. Number of generated beams

From the previous analysis, it is clear that, for a given chord error, LSLT generates a lighter triangulation than the other methods. Thus, for a given number of triangles, LSLT should triangulate much more beams than CDT and MCM. This has been tested and the results are provided in Table 3. The number of triangles has been indirectly fixed with the file size that should not exceed 2Gb to avoid the additive manufacturing machines malfunctions. Beam radii and chord errors are the same as in Table 1. Only the cell size has been modified to try to insert as many beams as possible in 2Gb.

		Bunny	T-fork	Dragon
Number of beams to reach 2Go	MCM	165800	170800	70000
	CDT	394378	822321	N/A
	LSLT	1614100	1462000	1655000
Gain LSLT wrt MCM		9.73	8.55	23.64
Gain LSLT wrt CDT		4.09	1.78	N/A

Table 3: Gains in terms of beams when comparing our LSLT technique to MCM and CDT.

Results clearly show that using our LSLT technique to triangulate lattice structures allows the insertion of much more beams for a given chord error. As a consequence, designers have more freedom in the definition of their lattice structures and notably on their resolution.

4.3. Triangles shape quality

Figure 20 shows the relative distribution of the triangles shape quality in percentage of the total number of triangles (vertical axis) per decile of quality (horizontal axis) when using MCM. Here again, triangles shape quality is defined as the ratio between its inner and outer circles [16]. First decile (decile 1) gathers together triangles close to equilateral, whereas the last decile (decile 10) corresponds to very elongated triangles. Five voxel grids have been tested while playing with the number of squares that can fit into a beam's section. For example, MCM Q5 corresponds to a voxel grid which size is so that 5 squares can fit into a beam's section. Clearly, for MCM, increasing the precision of the voxel grid does not affect the relative distribution of the triangles in term of quality.

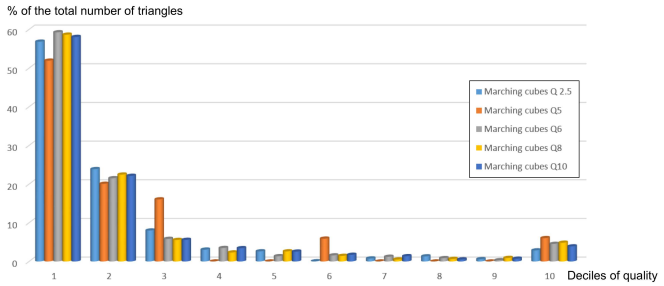

Figure 20: Evolution of the quality of the triangles when changing the size of the voxel grid (MCM).

For LSLT, the number of vertices per section affects the chord error, thus the accuracy, whereas the number of sections affects the quality of the triangles. Those two parameters are not correlated and can be dealt with separately. Figure 21 shows how triangles shape quality evolves when the number of intermediary sections increases. It clearly demonstrates that as the number of intermediary sections increases, the number of triangles in the first deciles increases. This is also visible in Figure 19. When the number of vertices per section increases from 6 to 12, the chord error decreases and the accuracy increases (Figure 19.b compared to 19.c). Similarly, when one more intermediary section is added, shape quality is improved (Figure 19.c compared to 19.d).

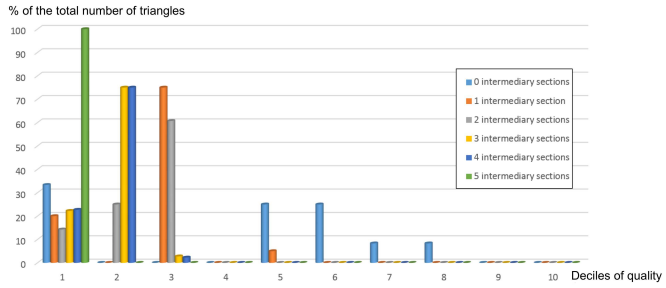

Figure 21: Evolution of the quality of the triangles when increasing the number of intermediary sections (LSLT).

As a conclusion, thanks to two independent control parameters, LSLT generates triangle meshes of better quality than MCM.

5. Conclusions and future works

Lattice Structure Lightweight Triangulation technique (LSLT) has been introduced and the different steps of the algorithm have been detailed. Contributions are threefold: (i) for a given chord error, our algorithm generates much less triangles than CDT and MCM, which allows for the insertion of much more beams in lattice structures; (ii) it is not sensible to the orientation of objects with respect to the reference frame; (iii) it is driven by two parameters which control separately the accuracy and the quality of the triangulation. Following our approach is much faster than constructing an exact model of a lattice

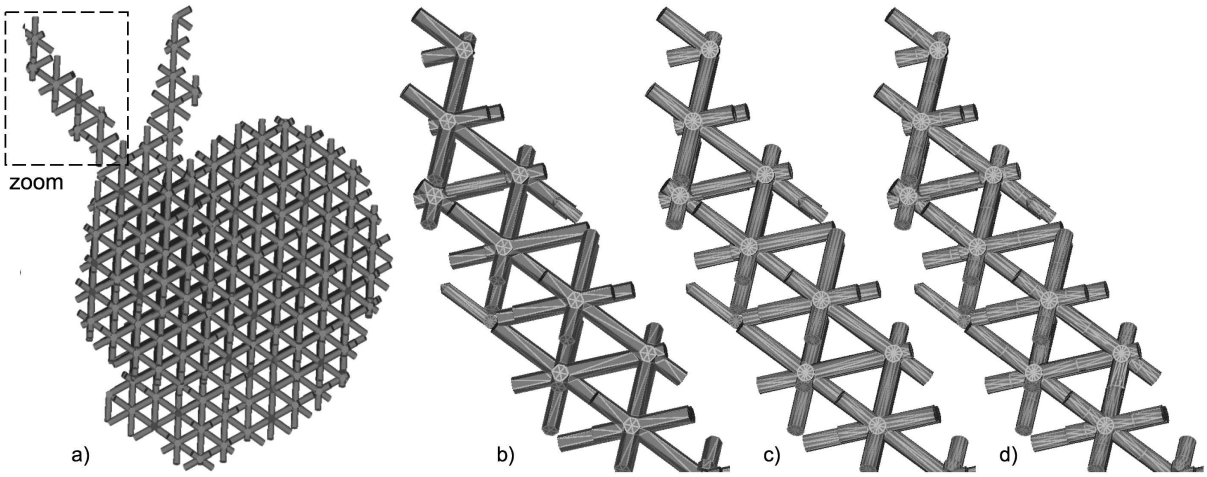

Figure 19: Evolution of the accuracy and quality of a triangulated lattice structure obtained from LSLT: (a) triangulated lattice structure, (b) with a chord error of 15% of the beam radius and no intermediary section, (c) with a chord error of 5% and no intermediary section, (d) with a chord error of 5% and one intermediary section.

structure and then tessellating it from its boundaries with CDT.

This proposed approach is also interesting when considering lattice structures optimization. The adopted data structure helps the manipulation of the lattice structure independently of its triangulation which is generally generated at the really end of the optimization process. Just as MCM, LSLT can be massively parallelized and thus lead to a very fast triangulation.

Moreover, the proposed approach can be pushed further to non uniform lattice structures. It is here proposed to contract and expand locally the lattice dimensions as to obtain overall 3D curvature for the lattice fibers as shown in Figure 22. Today, Poly-Shape company commonly uses the LSLT method to triangulate and build lattice structures.

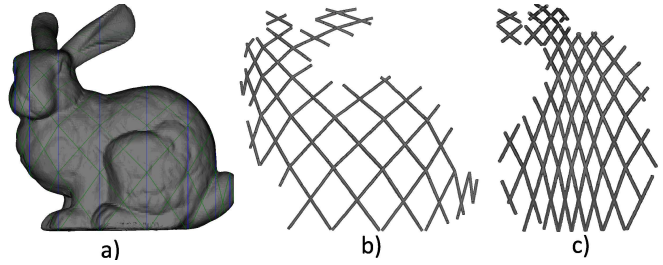

Figure 22: (a): 3D Graph of a non uniform lattice inserted within the Stanford bunny; (b): Front view of the triangulated mesh; (c): Side view of the triangulated mesh.

Finally, as it is, the proposed approach only applies to lattice structures that use beams as basic components. Some future work could lead to extend it to a more general case of lattice structures that would use more complex geometric features or shapes.

References

[1] Z. Qin, G. S. Jung, M. J. Kang, M. J. Buehler, The mechanics and design of a lightweight three-dimensional graphene assembly, Science advances 3 (1) (2017) –.

[2] W. Gao, Y. Zhang, D. Ramanujan, K. Ramani, Y. Chen, C. B. Williams, C. C. Wang, Y. C. Shin, S. Zhang, P. D. Zavattieri, The status, challenges, and future of additive manufacturing in engineering, Computer-Aided Design 69 (2015) 65–89.

[3] W. Regli, J. Rossignac, V. Shapiro, V. Srinivasan, The new frontiers in computational modeling of material structures, Computer-Aided Design 77 (2016) 73–85.

[4] L. P. Chew, Constrained delaunay triangulations, Algorithmica 4 (1) (1989) 97–108.

[5] S. Owen, A survey of unstructured mesh generation technology, Proceedings of the 7th International Meshing Roundtable. Sandia National Laboratories (1998) 239–267.

[6] W. E. Lorensen, H. E. Cline, Marching cubes: A high resolution 3d surface construction algorithm, SIGGRAPH Comput. Graph. 21 (4) (1987) 163–169.

[7] M. O. J.-H. C. Chien-Chang Ho, Bing-Yu Chen, Extended cubical marching squares for surface extraction from various kinds of volumetric structure, Computer-Aided Design (4).

[8] S. Raman, R. Wenger, Quality isosurface mesh generation using an extended marching cubes lookup table, Computer Graphics Forum 27 (3) (2008) 791–798.

[9] P. Young, T. Beresford-West, S. Coward, B. Notarberardino, B. Walker, A. Abdul-Aziz, An efficient approach to converting three-dimensional image data into highly accurate computational models, Philosophical Transaction of the Royal Society A. 366 (2008) 3155–3173.

[10] K. Ruohonen, Graph Theory, 2013.

[11] R. J. Wilson, Introduction to Graph Theory, 5th Edition, Prentice Hall/Pearson, 2010.

[12] A. Gray, E. Abbena, S. Salamon, Modern Differential Geometry of Curves and Surfaces with Mathematica, 3rd Edition, Chapman & Hall/CRC, 2006.

[13] R. Lou, J.-P. Pernot, A. Mikchevitch, P. Véron, Merging enriched finite element triangle meshes for fast prototyping of alternate solutions in the context of industrial maintenance, Computer-Aided Design 42 (8) (2010) 670–681.

[14] G. Mei, J. C. Tipper, Simple and robust boolean operations for triangulated surfaces, CoRR abs/1308.4434 (2013) –.

[15] S. Landier, Boolean operations on arbitrary polygonal and polyhedral meshes, Computer-Aided Design (85) (2017) 138–153.

[16] P. P. Pébay, T. J. Baker, Analysis of triangle quality measures, Mathematics of computation 72 (244) (2003) 1817–1839.

[17] W. Zhao, S. Gao, H. Lin, A robust hole-filling algorithm for triangular mesh, The Visual Computer 23 (12) (2007) 987–997.
[18] J.-P. Pernot, G. Moraru, P. Véron, Repairing triangle meshes built from scanned point cloud, Journal of Engineering Design 18 (5) (2007) 459–473.
[19] Y.-C. H. Lung-Chun Wang, Hole filling of triangular mesh segments using systematic grey prediction, Computer-Aided Design 44 (12) (2012) 1182–1189.