Carrier Baudouin *et al. Advanced Modeling and Simulation in Engineering Sciences* 2014, **1**:8
http://www.amses-journal.com/content/1/1/8

Advanced Modeling and Simulation in Engineering Sciences
a SpringerOpen Journal

RESEARCH ARTICLE Open Access

A frontal approach to hex-dominant mesh generation

Tristan Carrier Baudouin[1*], Jean-François Remacle[1], Emilie Marchandise[1], François Henrotte[1] and Christophe Geuzaine[2]

*Correspondence:
tristan.carrier@uclouvain.be
[1]Université catholique de Louvain, Institute of Mechanics, Materials and Civil Engineering, Bâtiment Euler, Avenue Georges Lemaître 4, Louvain-la-Neuve 1348, Belgium
Full list of author information is available at the end of the article

Abstract

Background: Indirect quad mesh generation methods rely on an initial triangular mesh. So called triangle-merge techniques are then used to recombine the triangles of the initial mesh into quadrilaterals. This way, high-quality full-quad meshes suitable for finite element calculations can be generated for arbitrary two-dimensional geometries.

Methods: In this paper, a similar indirect approach is applied to the three-dimensional case, i.e., a method to recombine tetrahedra into hexahedra. Contrary to the 2D case, a 100% recombination rate is seldom attained in 3D. Instead, part of the remaining tetrahedra are combined into prisms and pyramids, eventually yielding a mixed mesh. We show that the percentage of recombined hexahedra strongly depends on the location of the vertices in the initial 3D mesh. If the vertices are placed at random, less than 50% of the tetrahedra will be combined into hexahedra. In order to reach larger ratios, the vertices of the initial mesh need to be anticipatively organized into a lattice-like structure. This can be achieved with a frontal algorithm, which is applicable to both the two- and three-dimensional cases. The quality of the vertex alignment inside the volumes relies on the quality of the alignment on the surfaces. Once the vertex placement process is completed, the region is tetrahedralized with a Delaunay kernel. A maximum number of tetrahedra are then merged into hexahedra using the algorithm of Yamakawa-Shimada.

Results: Non-uniform mixed meshes obtained following our approach show a volumic percentage of hexahedra that usually exceeds 80%.

Conclusions: The execution times are reasonable. However, non-conformal quadrilateral faces adjacent to triangular faces are present in the final meshes.

Keywords: Advancing front methods; Tetrahedra recombination; Mixed hexahedral meshes

Background

Whether hex-meshing or tet-meshing is better for finite element computations is a long-standing controversy. This paper does not aim at deciding on that issue. Yet, it is a fact that a large number of finite element users would highly appreciate having automatic hex-meshing procedures for general 3D domains. A number of arguments can indeed be stated in favor of hex-meshing. For the same number of vertices, hex meshes have fewer elements, which speeds up the matrix/residual assembly. In solid mechanics, hexahedra exhibit higher accuracy than tetrahedra [1], which are plagued by locking problems [2].

In fluid dynamics, boundary layers made of hexahedra are effective for capturing large gradients and resolving viscous flows near the boundary, and semi-structured boundary-layer meshes attract significant interest (see, e.g. [3-5]).

Hexahedral mesh generation is still an ongoing research [6], and a major conclusion so far is that the generation of full-hex conforming meshes on arbitrary domains is beyond our reach nowadays. Relaxing a bit the requirements, hex-dominant meshes [7], in contrast with full-hex meshes, are allowed to aggregate a mixture of hexahedra, prisms, pyramids and tetrahedra. The goal of hex-dominant meshing is to generate meshes where hexahedral elements dominate, both in number and volume. This paper presents such an algorithm to automatically generate non-uniform isotropic hex-dominant meshes in arbitrary geometries. However, quadrilateral faces adjacent to triangular faces are usually found in the resulting meshes. Such non-conformities represent an additional complication for finite element methods. Various attempts at palliating the impact of these non-conformities have been discussed in the literature [8].

The proposed approach relies on an indirect strategy. The tetrahedra of an initial mesh are combined into hexahedra using Yamakawa-Shimada's algorithm [9], which works basically as follows: (*i*) All tetrahedra of an initial mesh are considered one after the other. (*ii*) The neighbors of each tetrahedron are visited in order to identify potential hexahedra. Candidate hexahedra are stored in an array and sorted with respect to their geometrical quality. (*iii*) The algorithm then iterates through this array, starting from the highest quality hexahedron, in order to effectively generate the hexahedral elements. Hexahedra that are composed of available tetrahedra (not marked for deletion) and that preserve hexahedral conformity are successively added to the mesh.

However, meshes obtained by applying a recombination algorithm to an arbitrary tetrahedral mesh fail to be hex-dominant. As an illustration, a mesh (depicted on Figure 1) was created using Yamakawa-Shimada's recombination algorithm from a tetrahedral mesh generated with the Delaunay refinement algorithm [10] of Gmsh [11]. The number of hexahedra in this mesh represents only 12.34% of the total number of elements.This low percentage of hexahedra results from the fact that the mesh vertices were not placed so as to favor recombination. In order to obtain higher ratios of good quality hexahedra, vertices need to be anticipatively aligned into a lattice-like structure that respects the user prescribed mesh size and the preferred directions of the mesh. This is what we are going to do with a specific frontal algorithm.

In a perfect hexahedral mesh, each interior vertex is linked with six other vertices: left-right, above-below, front-back (four vertices in case of a perfect quadrangular mesh). The main idea of our vertex placement algorithm is based on that observation. Knowing the prescribed local mesh size and the local preferred mesh directions, each interior vertex attempts to spawn six new vertices. A prospective vertex, however, is effectively created only if it lays inside the domain and if it is not too close to an existing vertex. This algorithm is applied to the boundaries of the geometry, prior to the volumes. When done, the vertices are tetrahedralized and Yamakawa-Shimada's algorithm can be applied.

Our approach has some similarities with the advancing front method. The vertices are created layer by layer toward the center of the geometry. However, contrary to the advancing front method, our algorithm does not construct a mesh topology along the way. All tetrahedra are built at the end. Figure 2 illustrates the various steps of our approach. The basic input is a CAD geometry file readable by the Gmsh free software.

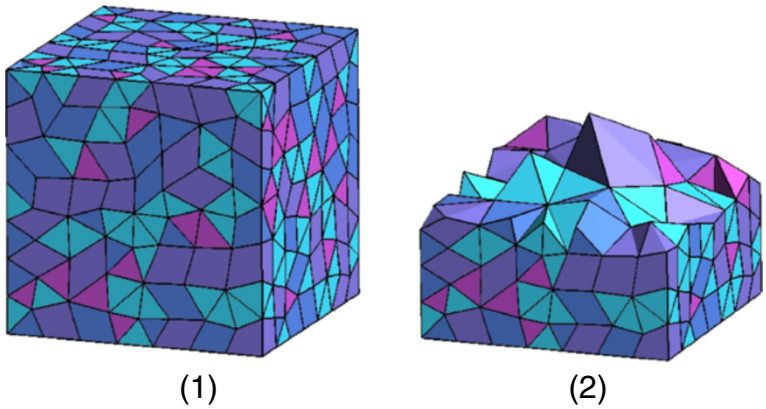

Figure 1 A mixed mesh obtained from an arbitrary tetrahedral mesh. Hexahedra are displayed in purple, prisms in dark blue, pyramids in pink and tetrahedra in light blue. **(2)** is a cutaway view of **(1)**.

The *Methods* section first examines more closely a number of data structures, tools and concepts that contribute to make the algorithm more efficient. It then describes the two- and three-dimensional versions of the frontal algorithm. Finally, it discusses Yamakawa-Shimada's algorithm and the issue of finite element conformity. The *Results and discussion* section presents a number of application examples, including mesh statistics and execution times.

Previous work

A large variety of procedures for hexahedral meshing have been proposed over the years. This section reviews a certain number of them, focusing on the automatic ones.

Several techniques extract hexahedral meshes from octree data structures [12]. These methods can generate non-uniform full-hex meshes in arbitrary geometries. The hexahedra orientation is defined by the octree root, which is a cube englobing the geometry. Boundary hexahedra perpendicular to the boundary surface can be achieved by projecting buffer layers [13,14]. The main limitations of octree-based approaches is that the generated hexahedra cannot be oriented flexibly, and that the quality of the hexahedra near boundaries is degraded [15].

Certain parametrization methods have been able to mesh complex three-dimensional domains [16,17]. These methods start by computing a three-dimensional direction field. The singularities of the direction field are then identified. Later on, the domain is cut in order to place all singularities on the boundary and to reduce the genus to 0 [18]. A parametrization minimizing the difference between the hexahedra orientation and the three-dimensional direction field is selected [19]. The supplementary cutting required to place all singularities on the boundary is necessary because it leads to a better parametrization [16]. Full hexahedral meshes of very good quality can be obtained for complicated domains. However, a variable mesh size cannot be prescribed [20]. Parametrization methods are very diverse. For example, a few algorithms capable of deforming a three-dimensional domain into a polycube model have been developed [21]. The polycube model is meshed and re-deformed back into its original shape.

Graph theory can be applied to the problem of creating hexahedra by recombining tetrahedra [22]. The starting point of the method is a tetrahedral mesh, which can

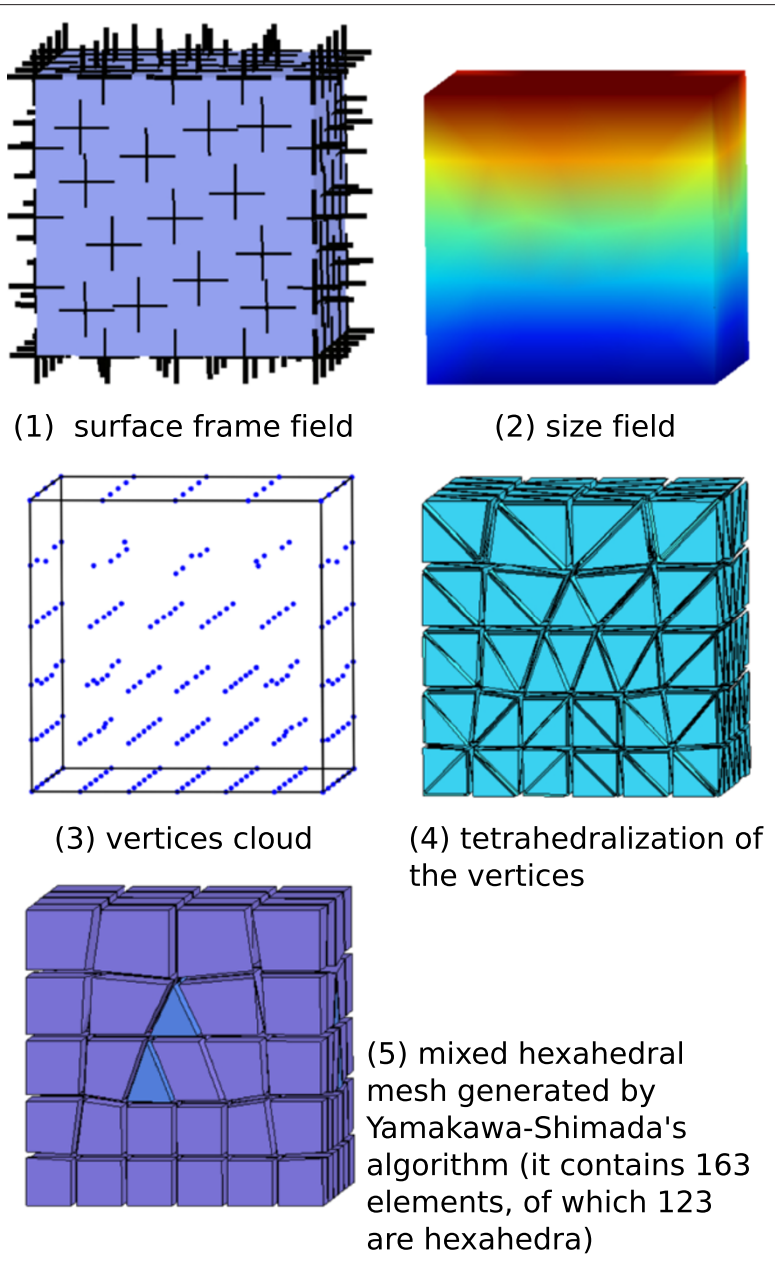

Figure 2 An overview of the steps necessary to generate a hex-dominant mesh.

be viewed as a graph. The method consists in searching through the mesh to identify subgraphs yielding hexahedra. There are six particular subgraphs to look for. Potential hexahedra can be constructed immediately or after classification along various criterions. This approach can be applied to prisms and pyramids as well. The recombination algorithm used throughout this article has been devised by Yamakawa and Shimada [9]. These authors have also designed an iterative procedure to align vertices in three dimensions. The first step of the procedure consists of filling the domain with crystal cells. Each crystal cell has the shape of a cube: there is one atom at the center and eight atoms in the corners. Crystal cells exert force on each other via their atoms. A set of equations of motion govern the cells positions. Throughout the process, cells can be added or removed

depending on the local density. Once cells velocities have sufficiently decrease, the center of each crystal cell becomes a mesh vertex.

To improve the spatial distribution of the set of vertices (which has a large impact on the effectiveness of such graph-based approaches), several techniques have been proposed, such as a modified version of Lloyd's algorithm. Lloyd's algorithm repeatedly moves vertices to the centroid of their Voronoi cell [23]. As the number of iterations increases, the Voronoi cells take the shape of perfect hexagons. In fact, it has been proven that Lloyd's algorithm minimizes an energy functional equal to the sum of the moments of inertia of the Voronoi cells [24]. Lévy-Liu's algorithm consists of minimizing higher order moments of inertia: the Voronoi cells then become squarely instead of hexagonal, which has the effect of aligning vertices in precise directions. Tetrahedral meshes smoothed by Lévy-Liu's algorithm make excellent candidates for recombination. When used in conjunction with the graph method discussed above, Lévy-Liu's algorithm can generate mixed meshes with high hexahedra percentages [24]. However, the method presented in this article is frontal, not iterative. The vertices are created layer by layer.

Methods

We have pointed out above the importance of having the mesh vertices pre-aligned to ensure a good recombination rate. It is the purpose of the *vertex placement algorithm* to achieve this. This algorithm, however, relies on a number of data structures and geometrical concepts that are first introduced and developed below.

First, the vertex placement algorithm needs to know, at each point of the domain, the prescribed local mesh size and the local preferred mesh directions. In practice, those geometrical quantities are conjointly obtained by evaluating a specific field structure called *cross field*. The generation of direction fields was extensively studied in [25].

Secondly, the notion of distance itself represents another degree of freedom of the method. We shall show that it is particularly appropriate when dealing with hex-meshing, to compute distances with the *infinity norm*, instead of the standard Euclidean norm.

Finally, the algorithm is characterized by a large number of spatial searches, in order to check whether or not a prospective vertices is too close to any already existing vertex. To optimize the efficiency of this operation, an *R-tree* data structure is employed [26,27].

Cross fields

At each point of a region $\Omega \subset \mathcal{R}^3$, the *frame field* $(\mathbf{d}_1, \mathbf{d}_2, \mathbf{d}_3)$ represents the three local orthogonal preferred directions of the hexahedral mesh. Frame fields are usually required to satisfy many constraints [16,28]. On the geometrical edges of Ω, one of the three directions should be tangent to the edge itself [9]. On the surfaces of Ω, one of the three directions should be perpendicular to the surface [9,16]. A last requirement is that the frame field should be as smooth as possible.

On the other hand, at each point of Ω, the *size field* represents the prescribed local mesh size value. Mesh sizes h_1, h_2, h_3 are defined for every point of the volume in each of the directions $\mathbf{d}_1, \mathbf{d}_2, \mathbf{d}_3$. In this paper, the mesh size field at a point $\mathbf{x}$ is isotropic, i.e. $h(\mathbf{x}) = h_1(\mathbf{x}) = h_2(\mathbf{x}) = h_3(\mathbf{x})$. The extension to anisotropic meshing will be done in a forthcoming work.

The user fixes the mesh size at the geometrical vertices of the model. One-dimensional size fields are then computed along the geometrical edges. Because the surfaces are

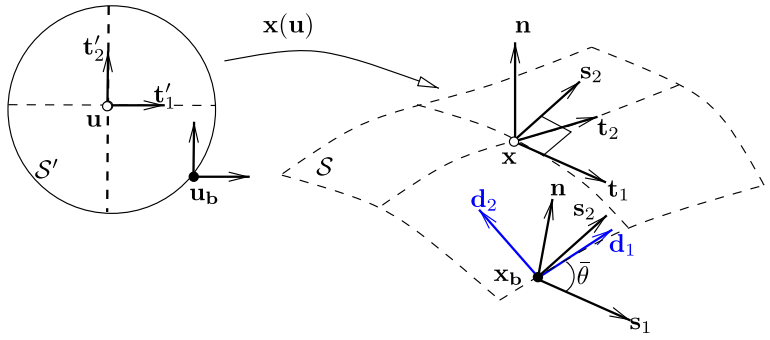

Figure 3 Surface parametrization and construction of the frame field (in blue).

bounded by geometrical edges, Dirichlet conditions can be imposed on the surfaces boundaries. A Laplace equation is used to obtain the size field over the surfaces. The size field over the volume is calculated in a similar manner. Continuous finite elements of the first order are employed in each case. The final size field is therefore a three-dimensional piecewise continuous field. The Laplace equation was chosen because it leads to smooth and gradual solutions.

The *cross field* $(h_1\mathbf{d}_1, h_2\mathbf{d}_2, h_3\mathbf{d}_3)$, now, combines both information into a single field. At each vertex of the mesh, the cross field evaluates into a symmetric real 3 by 3 tensor whose columns are the three orthogonal vectors parallel to the local preferred directions of the hexahedral mesh. Moreover, the norm of the vectors represent the local mesh size; the three norms are identical in case of an isotropic mesh (which is the case considered in this paper), but they may differ in case of an anisotropic mesh.

The construction of a frame field on a region Ω belongs to the category of elliptic problems. Boundary conditions must be imposed on the boundary $\partial\Omega$. We thus proceed logically by explaining first how the frame field is constructed on surfaces, and deal afterwards with the prolongation into the volumes.

Let

$$\mathbf{u} = \{u, v\} \in \mathcal{S}' \subset \mathcal{R}^2 \mapsto \mathbf{x}(\mathbf{u}) = \{x, y, z\} \in \mathcal{S} \subset \mathcal{R}^3, \tag{1}$$

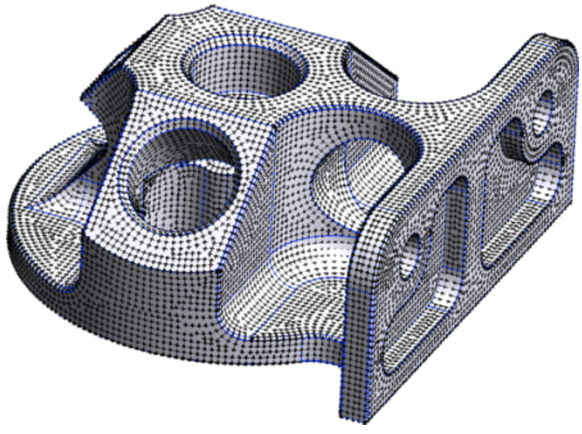

Figure 4 Cross fields on the surfaces of a mechanical part.

be a smooth parametrization of the surface $\mathcal{S}$ (see [29-31] for a review of parametrization techniques for surface remeshing). It should be noted that the parametrization does not need to be conformal, i.e the angles do not need to be conserved, for the algorithms presented in this paper. (This is a nice feature because guaranteed one-to-one conformal maps are more difficult to compute than bijective harmonic mapping.) For example, Figure 3 shows a harmonic parametrization of an arbitrary surface $\mathcal{S}$ onto a unit disk.

Consider the two tangent vectors

$$\mathbf{t}_1 = \frac{\partial \mathbf{x}}{\partial u} \text{ and } \mathbf{t}_2 = \frac{\partial \mathbf{x}}{\partial v},$$

which are the images in $\mathcal{S}$ of the basis vectors $\mathbf{t'}_1 = (1, 0)$ and $\mathbf{t'}_2 = (0, 1)$ of the parameter plane $\mathcal{S}'$. Because they are not parallel for any point of $\mathcal{S}'$, one can build the unit normal vector $\mathbf{n} = \mathbf{t}_1 \times \mathbf{t}_2 / \|\mathbf{t}_1 \times \mathbf{t}_2\|$. Each vector $\mathbf{t}$ tangent to $\mathcal{S}$ can be expressed as $\mathbf{t} = u\mathbf{t}_1 + v\mathbf{t}_2$ with (u, v) the covariant coordinates of $\mathbf{t}$. The tangent vector $\mathbf{t}$ is thus the image of a vector $\mathbf{t}' = (u, v)$ in the parameter plane. It is easy to compute covariant coordinates of any tangent vector $\mathbf{t}$ using the metric tensor of the parametrization. By definition, $\mathbf{t} = u\mathbf{t}_1 + v\mathbf{t}_2$. Then, $\mathbf{t} \cdot \mathbf{t}_1 = u\mathbf{t}_1 \cdot \mathbf{t}_1 + v\mathbf{t}_2 \cdot \mathbf{t}_1$ and $\mathbf{t} \cdot \mathbf{t}_2 = u\mathbf{t}_1 \cdot \mathbf{t}_2 + v\mathbf{t}_2 \cdot \mathbf{t}_2$, which reads in matrix form

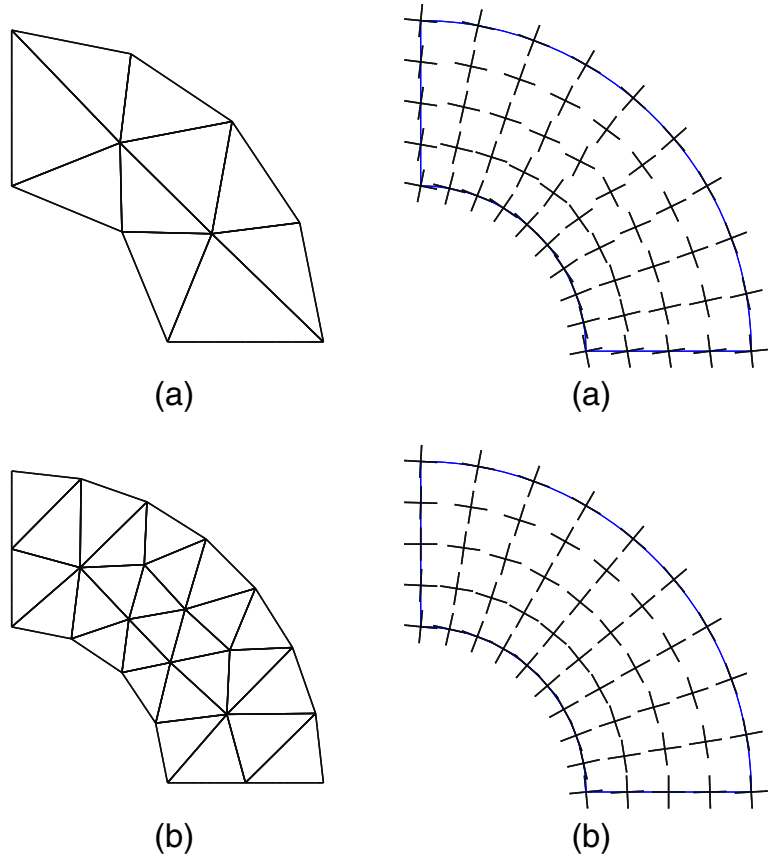

Figure 5 The mesh (a) was used to create the frame field (a) and the mesh (b) was used to create the frame field (b).

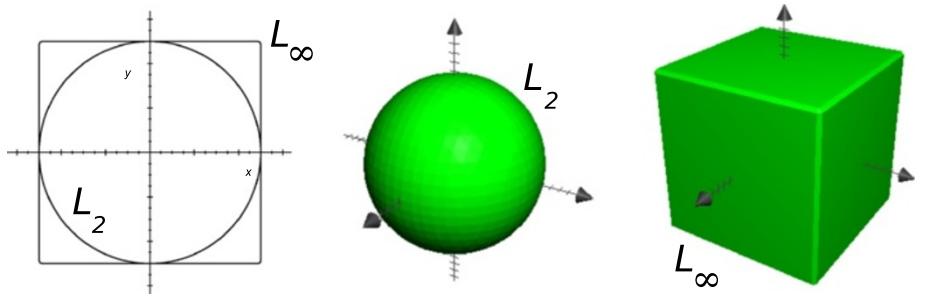

Figure 6 Unit circles and unit spheres in the L_2 and L_∞ norms.

$$\underbrace{\begin{bmatrix} \mathbf{t}_1 \cdot \mathbf{t}_1 & \mathbf{t}_1 \cdot \mathbf{t}_2 \\ \mathbf{t}_2 \cdot \mathbf{t}_1 & \mathbf{t}_2 \cdot \mathbf{t}_2 \end{bmatrix}}_{\mathcal{M}} \begin{bmatrix} u \\ v \end{bmatrix} = \begin{bmatrix} \mathbf{t} \cdot \mathbf{t}_1 \\ \mathbf{t} \cdot \mathbf{t}_2 \end{bmatrix} \tag{2}$$

where $\mathcal{M}$ is the metric tensor, invertible for any smooth parametrization.

For defining our frame field, a local orthonormal frame $(\mathbf{s}_1, \mathbf{s}_2, \mathbf{n})$ is first constructed at all points $\mathbf{x}$ of $\mathcal{S}$ with $\mathbf{s}_1 = \mathbf{t}_1/\|\mathbf{t}_1\|$, $\mathbf{s}_2 = \mathbf{n} \times \mathbf{t}_1/\|\mathbf{n} \times \mathbf{t}_1\|$. Next, the direction $\mathbf{d}_1$ of the frame field is computed at the points $\mathbf{x}_b$ of the boundaries of surface $\mathcal{S}$: $\mathbf{d}_1$ is the tangent vector to the boundary. The local orientation of the frame field $\bar{\theta}$ at the boundary can then be computed as the oriented angle between $\mathbf{s}_1$ and $\mathbf{d}_1$. Then, an elliptic boundary value problem is used to propagate the complex number $z(\mathbf{u}) = a(\mathbf{u}) + ib(\mathbf{u}) = e^{4i\theta(\mathbf{u})}$ in the parametric domain. More specifically, two Laplace equations with Dirichlet boundary conditions are solved in the parametric space $\mathcal{S}'$ in order to compute the real part $a(\mathbf{u}) = \cos 4\theta$ and the imaginary part $b(\mathbf{u}) = \sin 4\theta$ of z:

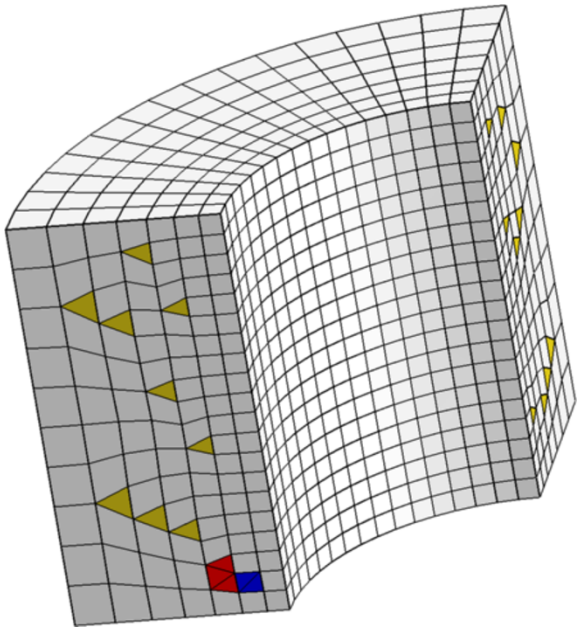

Figure 7 A mixed mesh created with the frontal algorithm using the L_∞ norm. The percentage of hexahedra by number is equal to 76.03% for L_∞ and 71.25% for L_2.

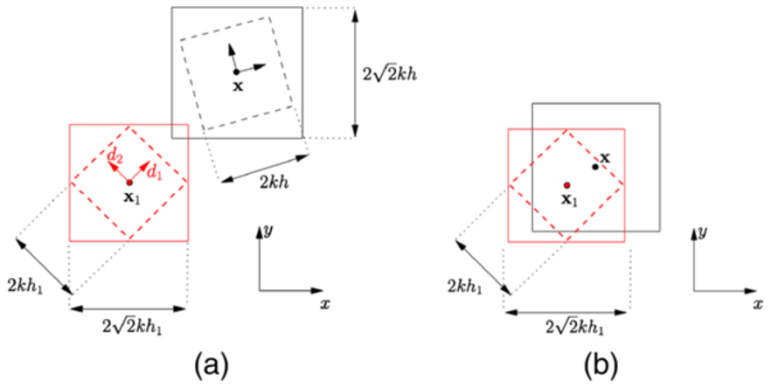

Figure 8 Two examples of bounding boxes intersections: a) $\mathbf{x}_1$ can be inserted in the cloud because $\mathbf{x}$ is outside the red dotted square b) $\mathbf{x}_1$ cannot be inserted in the cloud because $\mathbf{x}$ is inside the red dotted square.

$$\begin{aligned} &\nabla^2 a = 0, \quad \nabla^2 b = 0 \ \text{ on } \quad \mathcal{S}', \\ &a = \bar{a}(\mathbf{u}), \quad b = \bar{b}(\mathbf{u}) \ \text{ on } \quad \partial\mathcal{S}'. \end{aligned} \tag{3}$$

After solving those two PDEs, the frame field can be represented in the whole domain by the angle

$$\theta(\mathbf{u}) = \frac{1}{4}\text{atan2}(b(\mathbf{u}), a(\mathbf{u})).$$

The choice 4θ as the argument of z is motivated by symmetry arguments: frame fields are equivalent when they are rotated around $\mathbf{n}$ by any angle that is a multiple of $\pi/2$. Details of that procedure are given in [32]. Finally, the frame field $(\mathbf{d}_1, \mathbf{d}_2, \mathbf{d}_3)$ can be computed on the whole surface $\mathcal{S}$ as follows:

$$\mathbf{d}_1 = \mathbf{s}_1 \cos\theta + \mathbf{s}_2 \sin\theta, \quad \mathbf{d}_2 = -\mathbf{s}_1 \sin\theta + \mathbf{s}_2 \cos\theta, \quad \mathbf{d}_3 = \mathbf{d}_1 \times \mathbf{d}_2, \tag{4}$$

where θ is the solution of the elliptic boundary value problem (3).

As an example, Figure 4 presents the frame field computed on surfaces of a mechanical part. Figure 5 shows two triangular meshes of different coarseness and their resulting frame fields. Linear interpolation of the a and b components discussed earlier was used in

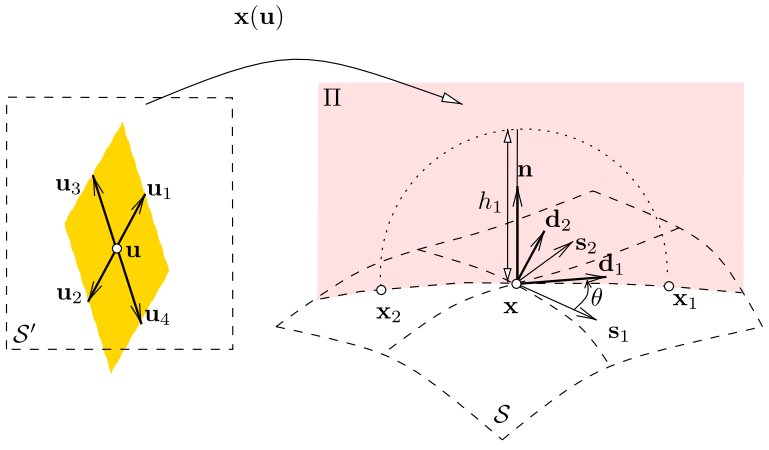

Figure 9 View of the different parameters of the point insertion scheme.

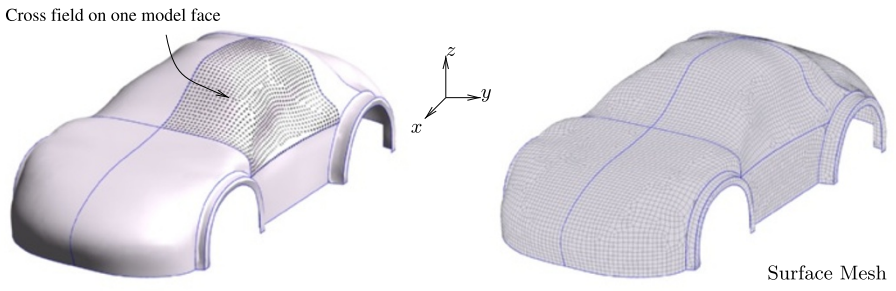

Figure 10 Different stages of the vertex insertion process. Left figure shows the cross field on one of the model faces of the car model. The right figure shows the quadrangular surface mesh that has been done using the points in the 3D space.

order to obtain the same number of frames regardless of the mesh density. As seen from the figure, the frame field (a) is not entirely radial and contains defects because the mesh (a) is too coarse.

The frame field at any point inside the volume is then chosen to be equal to the frame field at the closest surface vertex [24]. (ANN nearest neighbor library is employed for the queries [33].)

These frame fields are not going to be smooth whenever the distance function to the walls is not itself smooth. Recently, two methods capable of generating smooth frame fields have been developed [16,17]. Both of these methods employ LBFGS optimization to minimize energy functionals.

Measuring distances

For inserting a new mesh vertex in our frontal algorithm, the distance between a prospective vertex $\mathbf{x}_i$ and any already existing vertex $\mathbf{x}$ must be smaller than kh, where h is the local mesh size and k a free parameter of the algorithm ranging from 0 to 1. Parameter k absolutely needs to be inferior to one. If not, too many valid vertices will be missing from the cloud. In the implementation described in this work, k is equal to 0.7.

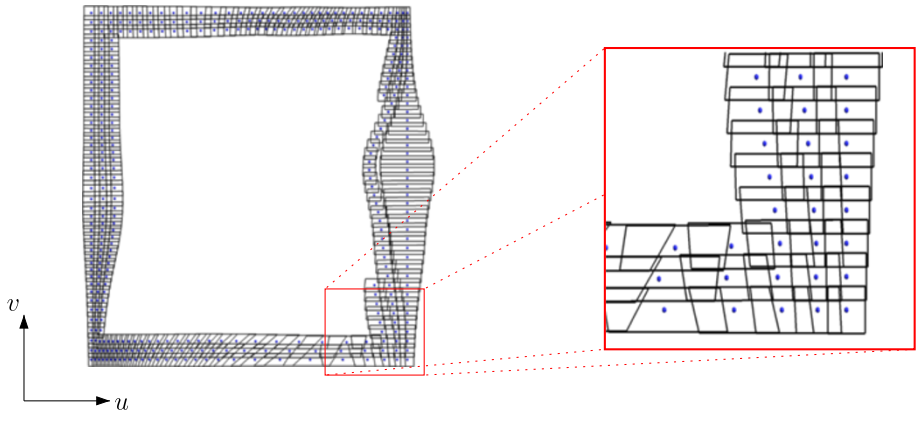

Figure 11 Vertex insertion process during the frontal packing of parallelograms algorithm. The figure shows points in the parametric *uv*-space with their respective exclusion parallelograms. Note that each parallelogram contains exactly one point.

The way distances between vertices are calculated is however a degree of freedom of the method. When dealing with hex-meshing, it turns out to be advantageous to compute distances in the *infinity norm*, instead of in the Euclidean norm:

$$||\mathbf{x}-\mathbf{y}||_2 = \sqrt{|x_1-y_1|^2+|x_2-y_2|^2+|x_3-y_3|^2} \tag{5}$$

$$||\mathbf{x}-\mathbf{y}||_\infty = \max(|x_1-y_1|, |x_2-y_2|, |x_3-y_3|). \tag{6}$$

In the infinity norm, the unit *sphere* is actually a cube, which reduces to a square in two dimensions (see Figure 6). The exclusion area around each prospective vertex is therefore a cube, resp. a square, which precisely matches the shape of the elements one wishes to build.

Contrary to the Euclidean norm, the infinity norm is not isotropic and, consequently, it has an orientation which is given by the frame field. In the parameter plane, due to the change of coordinates (1), the exclusion area is the parallelogram determined by

$$d_\infty^{orien}(\mathbf{x},\mathbf{y}) = ||M_{\mathbf{x}}(\mathbf{x}-\mathbf{y})||_\infty < kh. \tag{7}$$

where $M_{\mathbf{x}}$ is the Jacobian matrix of (1), evaluated at $\mathbf{x}$.

The infinity distance is not a differentiable function [24]. However, this is not an issue, because the frontal algorithm does not require the computation of distance derivatives.

Using the infinity distance instead of the Euclidean distance can increase the hexahedra percentage. The quarter cylinder illustrated on Figure 7 provides an example where an improvement by 5% of the ratio of hexahedra is observed, by simply using the L_∞ norm instead of the L_2 norm in the R-tree spatial search algorithm described in the next section.

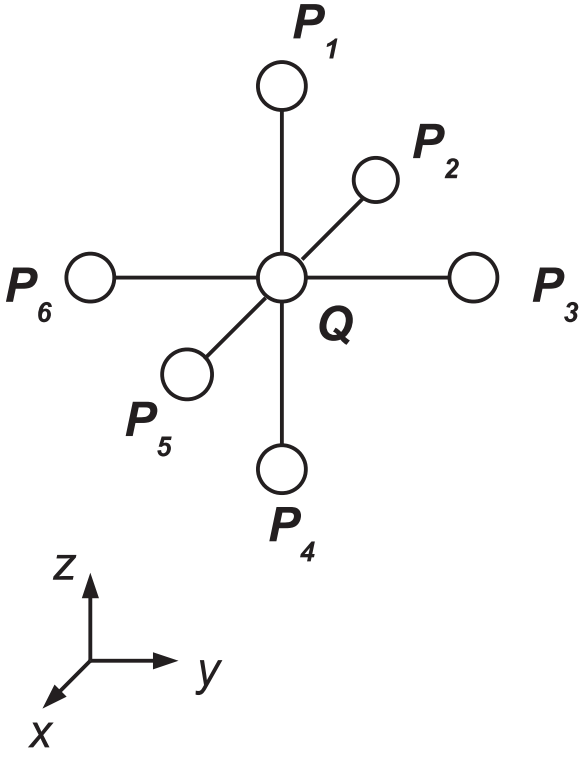

Figure 12 In three dimensions, vertex Q try to create six vertices: P_1, P_2, P_3, P_4, P_5 and P_6.

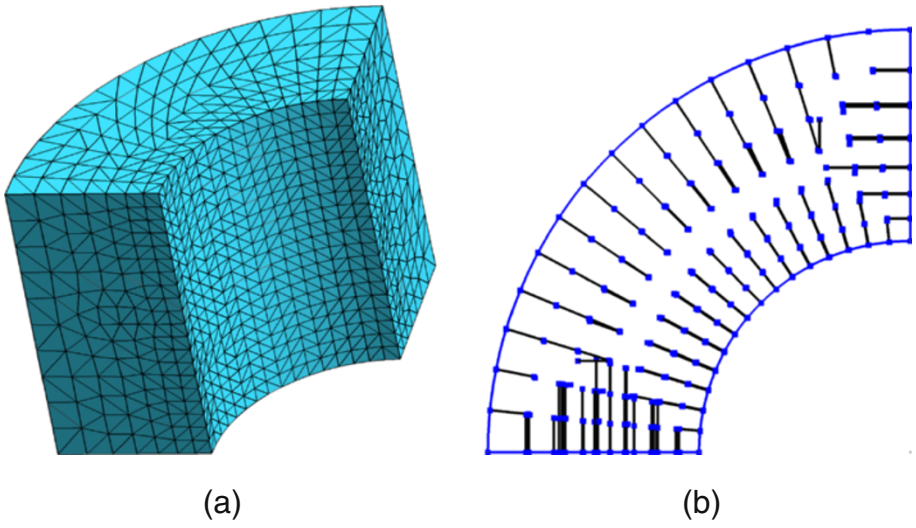

Figure 13 3D point insertion algorithm: (a) view of the surface mesh of a quarter cylinder, (b) top view of the quarter cylinder during the point insertion algorithm.

Using R-trees for spatial searches

As said before, a prospective vertex is effectively created only if there is enough unoccupied space around it. The size of this exclusion area or volume depends on the local mesh size. According to the dimension and the chosen norm, the shape of the exclusion region can be a parallelogram or an ellipse (in 2D), and a cube or a sphere (in 3D).

The computation of the distance between the prospective vertex and all the other vertices would have a quadratic complexity in time and would therefore be prohibitive in terms of computation time. The number of computations required to ensure the exclusions can however be considerably decreased if the exclusion cube of each vertex is enclosed in a bounding box whose edges are parallel to the coordinate axis. An R-tree

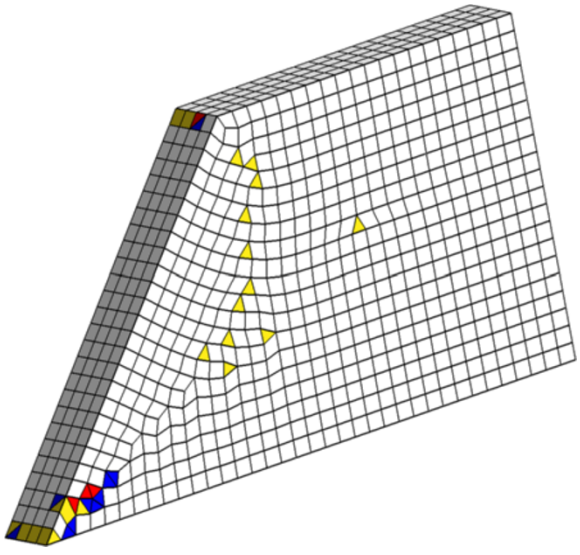

Figure 14 A mixed mesh containing hexahedra (white), prisms (yellow), pyramids (red) and tetrahedra (blue).

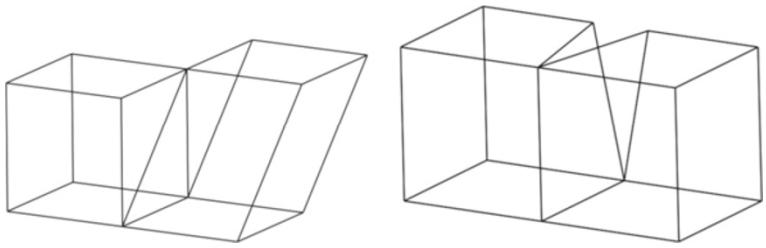

Figure 15 Non-conformities between two hexahedra sharing three vertices.

data structure [26,27] can efficiently determine bounding boxes intersections and, then, it is enough to compute the distance between pairs of vertices whose boxes intersect each other.

We now illustrate with a 2D example of a planar surface how to decide wether a prospective vertex can be inserted or not. For this example, we have chosen the infinity norm for computing distances. In Figure 8(a), $\mathbf{x}_1$ is the prospective vertex and $\mathbf{x}$ is an existing mesh vertex. The dotted square around $\mathbf{x}_1$ is the oriented exclusion area of vertex $\mathbf{x}_1$, that is computed from the surface cross field $(h_1\mathbf{d}_1, h_1\mathbf{d}_2)$ that has a uniform mesh size field h_1. The solid box surrounding the prospective vertex is the bounding box of the exclusion area that is parallel to the *xy*-coordinate axis. This bounding box should always include the oriented exclusion square of side $2kh$. This condition is satisfied in 2D if the box is of side $2\sqrt{2}kh$ and in 3D if the cube is of side $2\sqrt{3}kh$. Even if the boxes intersect each other in Figure 8(a), the distance between $\mathbf{x}_1$ and $\mathbf{x}$ is sufficiently large. Thus, $\mathbf{x}_1$ can be inserted in the cloud and added to the queue.

Figure 8(b) shows the same two vertices. Again, the boxes intersect each other. This time, however, $\mathbf{x}_1$ is too close to $\mathbf{x}$ and $\mathbf{x}_1$ cannot be added to the cloud or to the queue.

It should be noted that on Figures 8(a) and 8(b), $d_\infty^{orien}(\mathbf{x}_1, \mathbf{x})$ is not necessarily equal to $d_\infty^{orien}(\mathbf{x}, \mathbf{x}_1)$. The local mesh sizes at $\mathbf{x}_1$ and $\mathbf{x}$ can also be different as illustrated in Figure 8(a). However, if $\mathbf{x}$ is outside the dotted square of $\mathbf{x}_1$, it is considered sufficient.

For non-planar surfaces, the surfaces need to be parametrized. As the parametrization is not necessarily conformal, i.e. the angles between $\mathbf{d}_1$ and $\mathbf{d}_2$ are not conserved, the dotted squares (exclusion area) of Figure 8 become parallelograms in the parametric space. As far as the bounding boxes are concerned, they are computed in the same manner and are then parallel to the *uv*-coordinate axis of the parametric space.

Let's assume that on surfaces, each vertex attempts to create four vertices in the four cardinal directions. If the surface normal is not constant, these prospective vertices may

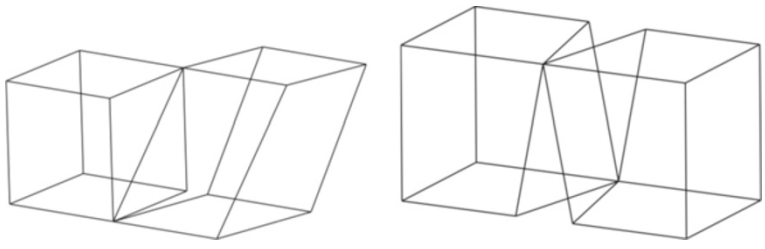

Figure 16 Non-conformities between two hexahedra sharing two vertices.

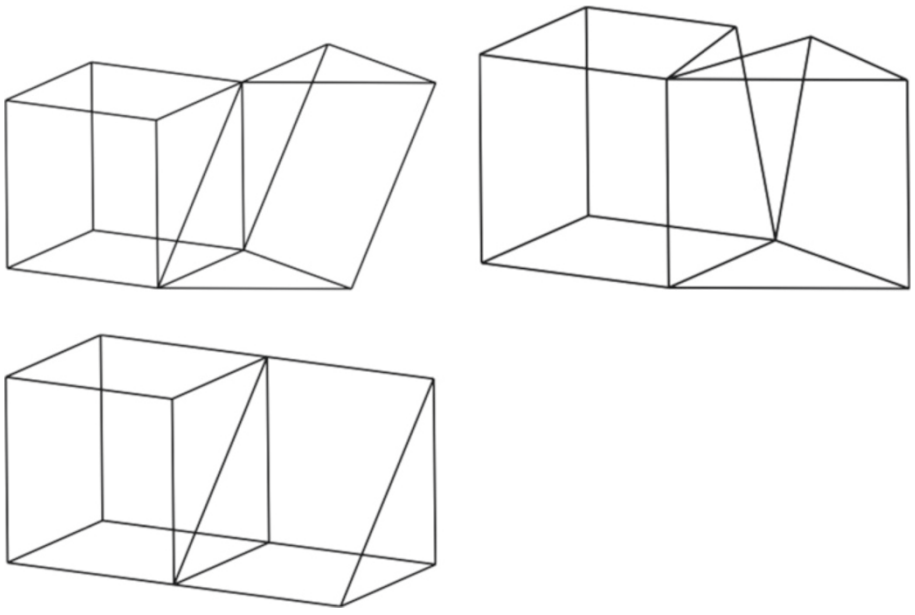

Figure 17 Non-conformities between one hexahedra and one prism sharing three vertices.

not rest on the surface. The next section describes a scheme capable of solving this issue by intersecting surfaces with circle arcs.

Surface meshing: the packing of parallelograms algorithm

The quadrilateral mesh algorithm presented here is a simpler variant of [32] that we call *packing of parallelograms*. Consider one vertex located at point $\mathbf{u} = (u, v)$ of the parameter plane $\mathcal{S}'$ which correspond to point $\mathbf{x}(u, v)$ in the 3D space (see Figure 9). The cross field at this point of the surface is $(h_1\mathbf{d}_1, h_2\mathbf{d}_2, h_n\mathbf{n})$, in terms of the three orthonormal preferred mesh directions, $\{\mathbf{d}_1, \mathbf{d}_2, \mathbf{n}\}$, and the three corresponding mesh sizes, $\{h_1, h_2, h_n\}$,.

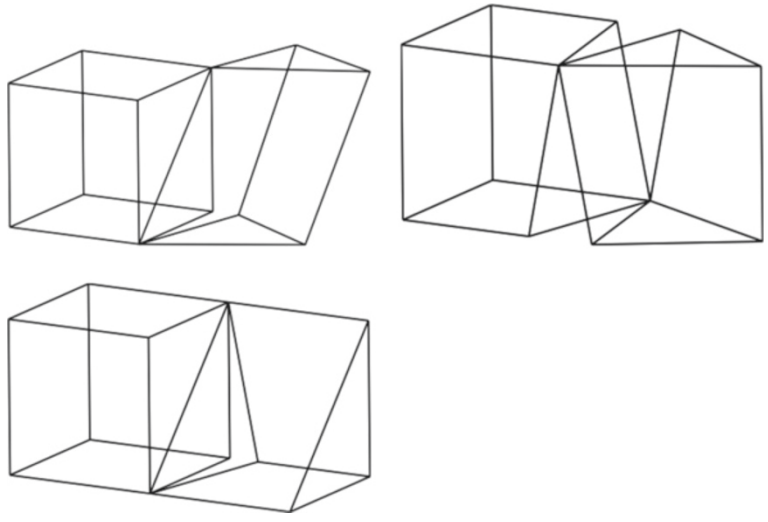

Figure 18 Non-conformities between one hexahedra and one prism sharing two vertices.

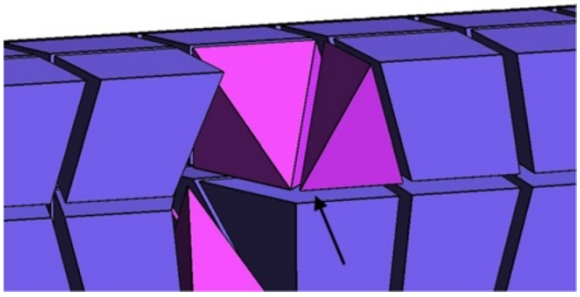

Figure 19 The arrow indicates a non-conformal face between a hexahedron and two pyramids. Tetrahedra are not shown to facilitate visibility.

In a perfect quad mesh, each vertex is connected to four neighboring vertices forming a cross parallel to the cross field. In our approach, four prospective points $\mathbf{x}_i,\ i = 1, \ldots, 4$ are constructed in the neighborhood of point $\mathbf{x}$ with the aim of generating the perfect situation.

Points $\mathbf{x}_1$ and $\mathbf{x}_2$ are constructed as the intersection of the surface $\mathcal{S}$ with a circle of radius h_1, centered on $\mathbf{x}$ and situated in the plane Π of normal $\mathbf{d}_2$ (see Figure 9). Points $\mathbf{x}_3$ and $\mathbf{x}_4$ are constructed as the intersection of the surface $\mathcal{S}$ with a circle of radius h_2, centered on $\mathbf{x}$ and situated in the plane of normal $\mathbf{d}_1$ (not in the figure for clarity).

Numerical difficulties associated with the surface-curve intersection are overcome by choosing a good initial guess for the intersection. If we approximate the surface by its tangent plane at $\mathbf{x}$, point $\mathbf{x}_1$ is situated at $\mathbf{x}_1 = \mathbf{x} + h_1\mathbf{d}_1$. A good initial guess in the parameter plane is $\mathbf{u}_1 = \mathbf{u} + d\mathbf{u}_1$ where $d\mathbf{u}_1 = (du_1, dv_1)$ is computed using (2) i.e.

$$\mathcal{M}\begin{pmatrix} du_1 \\ dv_1 \end{pmatrix} = \begin{pmatrix} h_1\mathbf{d}_1 \cdot \mathbf{t}_1 \\ h_1\mathbf{d}_1 \cdot \mathbf{t}_2 \end{pmatrix}.$$

This also gives $\mathbf{u}_2 = \mathbf{u} - d\mathbf{u}_1$, $d\mathbf{u}_3 = (du_3, dv_3)$

$$\mathcal{M}\begin{pmatrix} du_3 \\ dv_3 \end{pmatrix} = \begin{pmatrix} h_2\mathbf{d}_2 \cdot \mathbf{t}_1 \\ h_2\mathbf{d}_2 \cdot \mathbf{t}_2 \end{pmatrix}.$$

$\mathbf{u}_3 = \mathbf{u} + d\mathbf{u}_3$ and $\mathbf{u}_4 = \mathbf{u} - d\mathbf{u}_3$.

The algorithm works as follows. Each vertex of the boundary is inserted in a fifo queue. Then, the vertex $\mathbf{x}$ at the head of the queue is removed and its four prospective neighbors

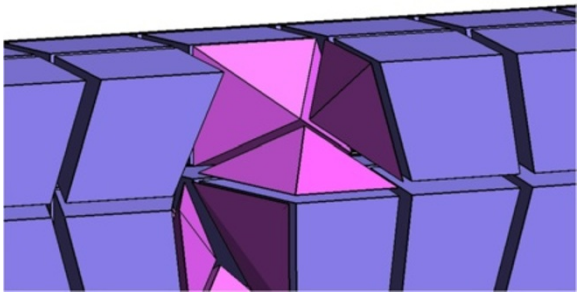

Figure 20 A supplementary pyramid was created by Owen-Canann-Saigal's algorithm to fix the non-conformity.

Table 1 Mesh data

Figure	# vertices	H_{nbr}	H_{vol}	Q	CPU (s)	NC
21	126,922	62.91%	85.70%	0.95	488 s.	7.19%
23	42,263	75.42%	92.85%	0.96	164 s.	3.97%
26	11,648	58.16%	82.68%	0.94	112 s.	7.70%
27	2737	60.23%	83.02%	0.94	45 s.	8.04%
28	6006	84.54%	94.10%	0.96	80 s.	2.25%
32	19,284	86.19%	94.93%	0.98	51 s.	1.81%

Lévy-Liu's algorithm [24,36] was used for Figures 26, 27 and 28.

$\mathbf{x}_i$ are computed. A new vertex $\mathbf{x}_i$ is inserted at the tail of the queue if the following conditions are satisfied: (*i*) vertex $\mathbf{x}_i$ is inside the domain and (*ii*) vertex $\mathbf{x}_i$ is not too close to any of the vertices that have already been inserted.

As for the first condition, it is enough to check if the preimage $\mathbf{u}_i \in \mathcal{S}'$ of $\mathbf{x}_i$ is inside the bounds of the parameter domain. Concerning the second condition, the distances on the surface $\mathcal{S}$ should theoretically be measured in terms of geodesics, This is however clearly overkill from a mesh generation point of view. We define an exclusion zone for every vertex that has already been inserted (this includes boundary vertices). This exclusion zone is a parallelogram in the parameter plane (see the yellow parallelogram of Figure 9). This parallelogram is scaled down by a factor $k = 0.7$ in order to allow the insertion of (at least) points x_i. The different stages of the procedure for a non planar surface are presented on Figure 10 and Figure 11. Then, the surfaces are triangulated in the parameter plane

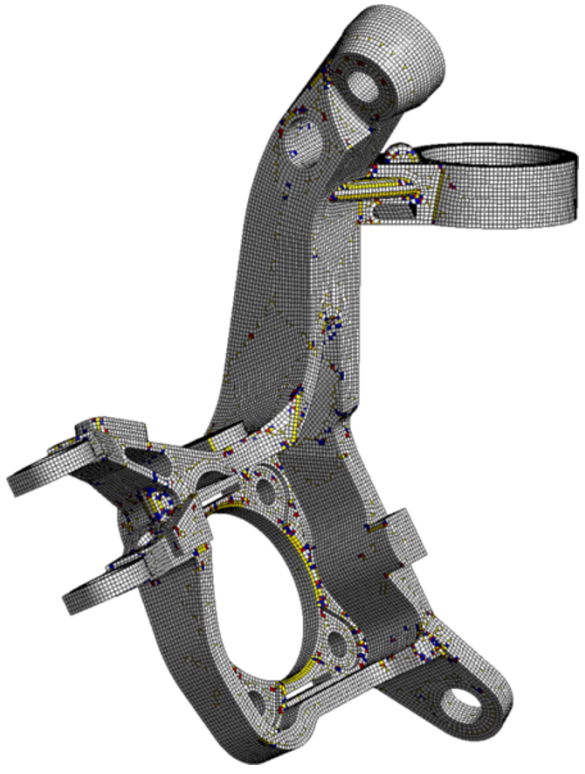

Figure 21 A mechanical piece.

using an anisotropic Delaunay kernel and the triangles are subsequently recombined into quadrilaterals using the Blossom-Quad algorithm [34].

As shown on Figure 11, exclusion areas can become anisotropic parallelograms in the parametric plane. However, they always correspond to squares in the three-dimensional space. The vertices are triangulated in the parametric plane. Anisotropic triangulation is therefore necessary in order to obtain the expected arrangement of right triangles.

Volume meshing: the 3D point insertion algorithm

Volume meshing proceeds in the same way as surface meshing. The procedure starts from a 2D triangular mesh that has been created using surfacic frame fields. A frontal algorithm is used to create well aligned vertices inside the volume, starting from surface points. The 3D point insertion algorithm works in the same manner as the one used for surfaces.

All boundary mesh vertices are initially pushed into a queue. The vertices are popped in order: each vertex $\mathbf{Q}$ popped out of the queue attempts to create six neighboring vertices in the six cardinal directions $\mathbf{P}_{1,2} = \mathbf{Q} \pm h\mathbf{d}_1$, $\mathbf{P}_{3,4} = \mathbf{Q} \pm h\mathbf{d}_2$, $\mathbf{P}_{5,6} = \mathbf{Q} \pm h\mathbf{d}_3$ at a distance h from itself (see Figure 12).

A prospective vertex is added to the vertices cloud and to the queue only if it satisfies the two following conditions:

1. It is inside the domain.
2. It is not too close to an existing mesh vertex, i.e. if the distance is smaller than kh.

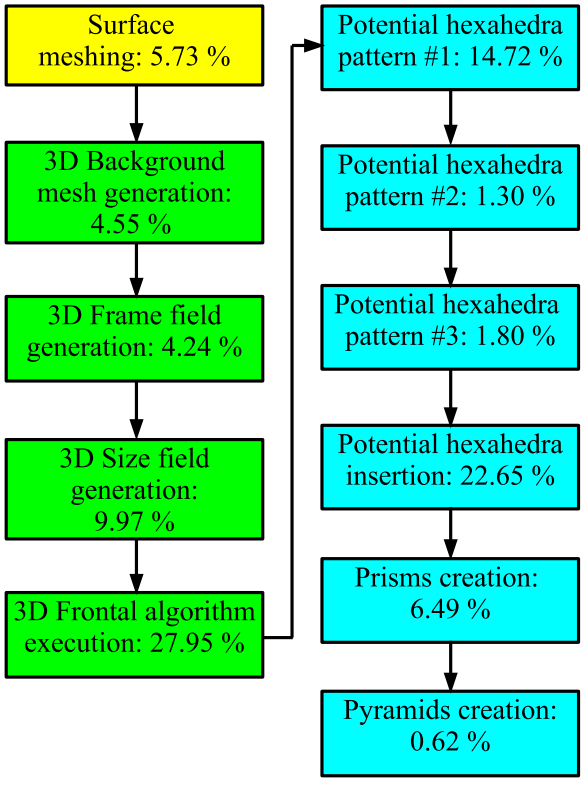

Figure 22 Global algorithm flowchart and execution time percentages.

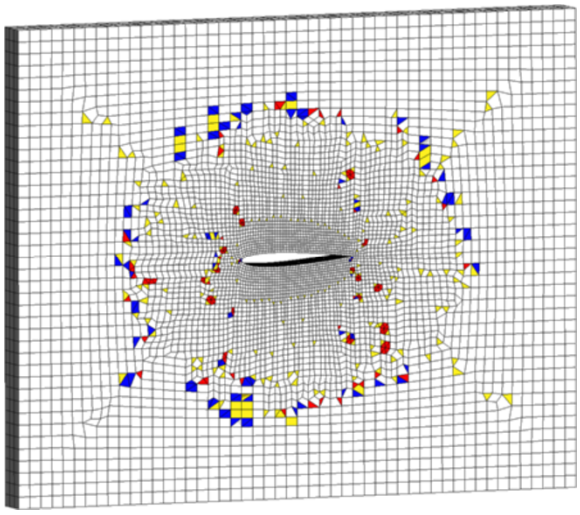

Figure 23 A NACA airfoil.

An octree data structure is again employed to efficiently determine if a vertex is inside the domain [11].

Eventually, no more prospective vertices can be added to the cloud without being too close to existing ones. The process then stops and the cloud is tetrahedralized with a Delaunay procedure [35].

The frontal algorithm was applied to the quarter cylinder starting from the surface mesh shown in Figure 13(a). In Figure 13(b), lines are traced between each vertex and its parent in order to observe the progression of the 3D point insertion algorithm.

The quality of the alignment inside the geometry is very dependent on the quality of the alignment on the boundaries. If the triangles on the boundaries are far from being right-angled, then the vertices inside the geometry will not be well aligned. Various algorithms are capable of generating sets of aligned vertices on surfaces, such as the Delquad

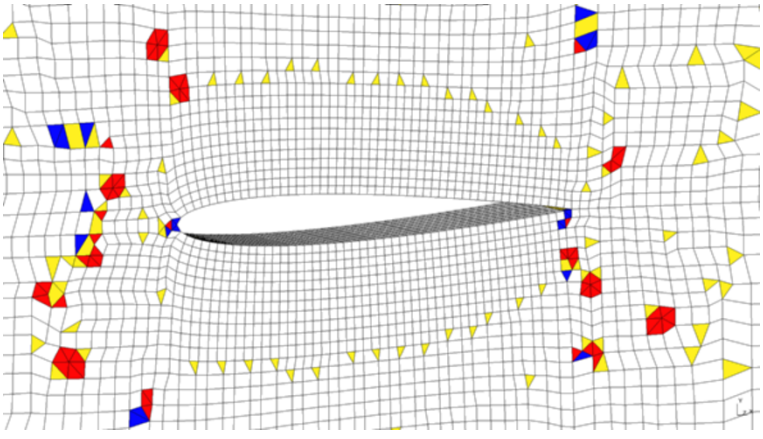

Figure 24 A zoom of the airfoil.

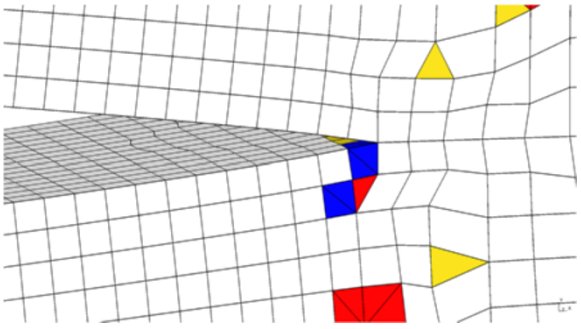

Figure 25 A zoom of the trailing edge.

algorithm [32] or Lévy-Liu's algorithm [24,36]. However, for the majority of the examples presented in this article, a two-dimensional version of the frontal algorithm was employed.

As explained earlier, each vertex attempts to create six other vertices at a distance $d = h$ from itself. For smoother size transitions, d can instead be an average between the local mesh size at the parent vertex and the local mesh size at the prospective vertex.

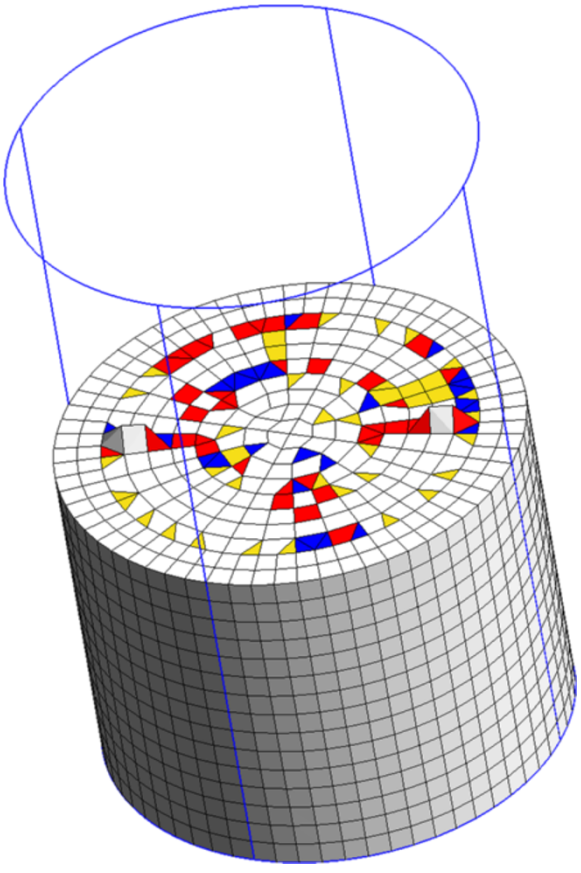

Figure 26 A cylinder. The blue lines illustrate the cylinder geometry.

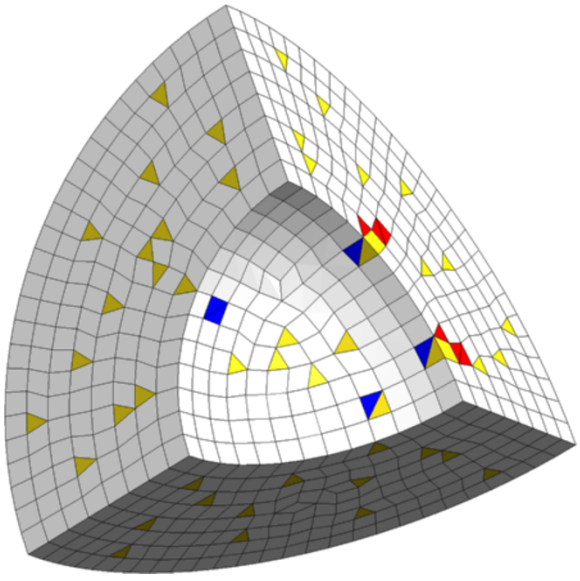

Figure 27 An hollow eighth of ball with a constant mesh size.

Volume meshing: Yamakawa-Shimada's algorithm and finite element conformity

This section briefly describes Yamakawa-Shimada's recombination algorithm. It then discusses the problem of finite element conformity in the case of mixed hex meshes.

Yamakawa-Shimada's algorithm begins by iterating through the tetrahedra of the initial mesh. For each tetrahedron, it attempts to find neighboring tetrahedra with which to construct a hexahedron. Five, six or seven tetrahedra are required to construct one hexahedron. Three patterns of assembly are considered. Two out of these three patterns are described in [9]. When a potential hexahedron is found, it is added to an array. However,

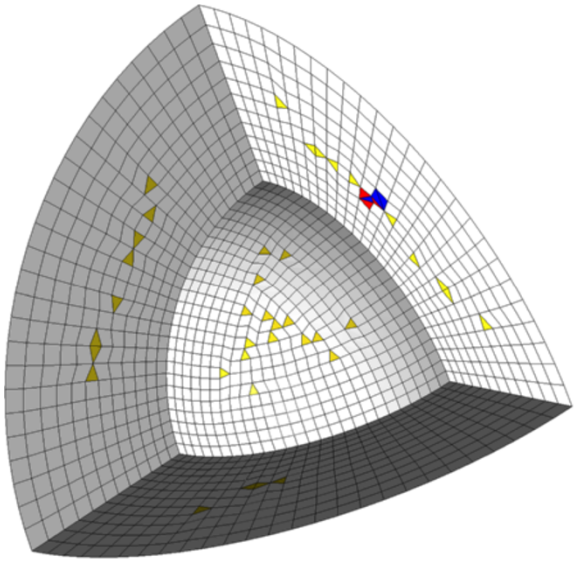

Figure 28 An hollow eighth of ball with a variable mesh size.

the hexahedron will not necessarily be part of the final mesh. Once all tetrahedra have been visited, the array is sorted by hex quality.

The quality Q is defined as follows:

$$Q = \min_{i=1..8} \left| \frac{\mathbf{v_{i1}} \cdot (\mathbf{v_{i2}} \times \mathbf{v_{i3}})}{||\mathbf{v_{i1}}|| \, ||\mathbf{v_{i2}}|| \, ||\mathbf{v_{i3}}||} \right|, \tag{8}$$

where i is the vertex number. For a hexahedron, i goes from 1 to 8; $\mathbf{v_{i1}}$, $\mathbf{v_{i2}}$ and $\mathbf{v_{i3}}$ are the three vectors parallel to the three edges connected to vertex i. Q is in fact the modulus of the minimum scaled Jacobian [9]. Evidently, Q is meaningless for invalid hexahedra. Invalid hexahedra are characterized by a null or negative Jacobian determinant, which renders the mesh improper for calculations.

Starting from the highest quality hexahedron, the algorithm then iterates through the array. Potential hexahedra composed of tetrahedra not yet marked for deletion are added to the mesh. The tetrahedra of the added hexahedron are then marked for deletion. It is to be noted that only a small fraction of potential hexahedra appear in the final mesh.

Prisms can later be added by following a similar procedure [9]. All prisms are composed of three tetrahedra. There is only one pattern of construction for prisms [9].

Figure 14 shows a mixed mesh created with Yamakawa-Shimada's algorithm.

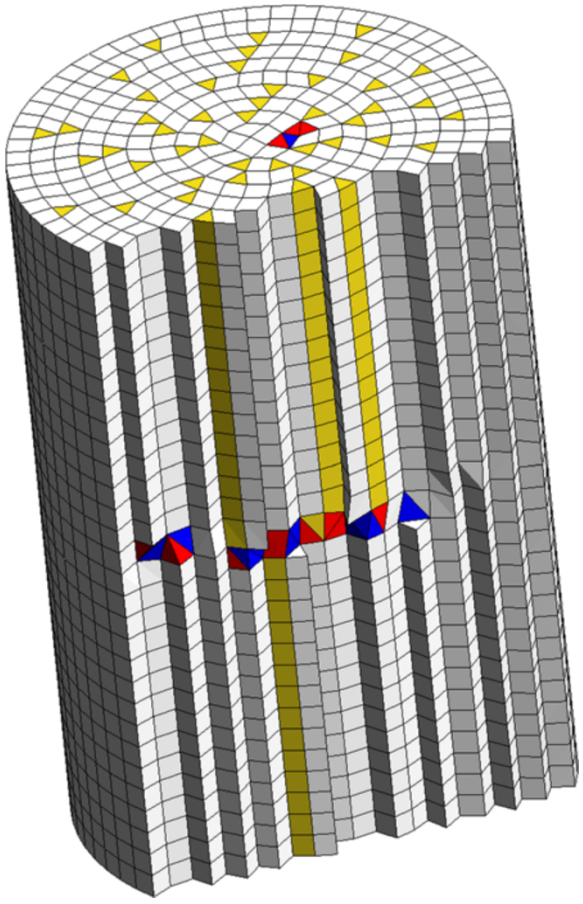

Figure 29 In this mesh, the front was initiated only from the lower and upper disks. The collision is visible in the middle of the cylinder.

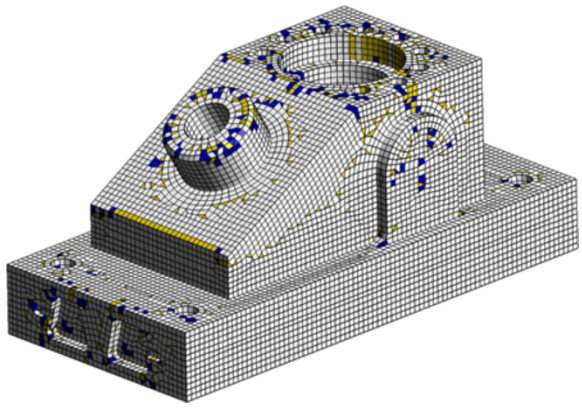

Figure 30 A mixed hexahedral mesh of the anc101 mechanical part.

Let's assume that finite elements of the first order are employed. The tetrahedral shape functions are therefore linear, while the hexahedral shape functions are trilinear [37]. On triangular faces, the interpolation is linear and takes into account three degrees of liberty. On quadrilateral faces, the interpolation is bilinear and takes into account four degrees of liberty [8,9]. If a nonplanar quadrilateral face is adjacent to a triangular face, there will be a gap or overlap [9]. The elements are not going to be a perfect partition of the domain anymore, which goes against the basic assumptions of the finite element method. Gaps or overlaps can also be created by several configurations of neighboring hexahedra or prisms. Figures 15 and 16 show four cases of non-conformities between hexahedra [9].

Figures 17 and 18 show six cases of non-conformities between one hexahedron and one prism. Non-conformities resulting from neighboring prisms can be deduced from these six cases.

Yamakawa-Shimada's algorithm should therefore avoid creating the configurations illustrated on Figures 15, 16, 17 and 18 while iterating through the sorted arrays of potential hexahedra and prisms. Non-conformities can be efficiently identified by employing hashing techniques.

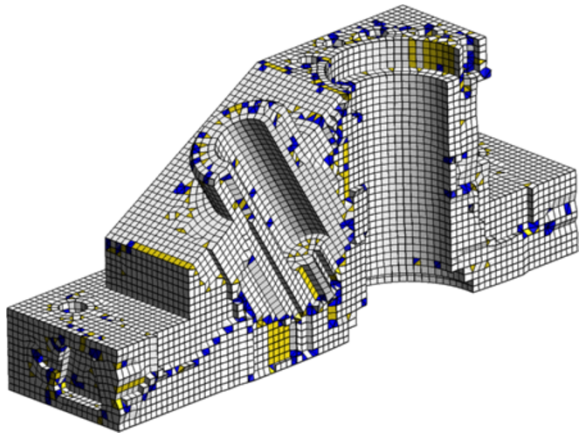

Figure 31 A cutaway view of the anc101 mesh.

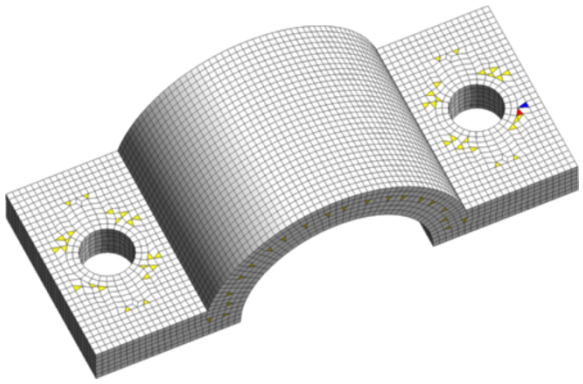

Figure 32 One of the mesh used for the convergence analysis. The mesh size is equal to $h = 1.0$.

After the creation of hexahedra and prisms, some tetrahedra are recombined into pyramids. Every pair of tetrahedra resting on a quadrilateral face is merged to form a pyramid. This step can fix many non-conformities. However, it does not resolve them all. As shown on Figure 19, many quadrilateral faces can still be adjacent to triangular faces belonging to either tetrahedra or pyramids.

These non-conformities can be fixed by Owen-Canann-Saigal's algorithm [38]. Owen-Canann-Saigal's algorithm first creates a flat pyramid on each non-conformal quadrilateral face. The apex of the pyramid is not initially present in the mesh, but it is added by the algorithm. Surrounding tetrahedra and pyramids need to be subdivided to accommodate this new vertex. The pyramid is then raised so it does not have a null volume. Figure 20 illustrates the pyramid constructed to correct the non-conformity on Figure 19.

Owen-Canann-Saigal's algorithm can render a mixed hexahedral mesh completely conformal. However, it has a drawback. It increases the number of tetrahedra and pyramids, which lowers the percentage of hexahedra by number. As a consequence, Owen-Canann-Saigal's algorithm was not used for the results presented below.

Some quadrilateral faces will be adjacent to one or two triangles. Finite element solvers capable of handling these type of non-conformities are required.

Results and discussion

This section presents several mixed hex meshes created with the frontal algorithm and Yamakawa-Shimada's algorithm. Three quantities are used to evaluate the quality of the meshes: the percentage of hexahedra by number H_{nbr}, the percentage of hexahedra by volume H_{vol} and the average hex quality Q defined in Eq. 8. In general, H_{vol} is higher than H_{nbr}. *CPU* designates the total execution time (in s) on a 2010 laptop computer. All the data is compiled in Table 1. The following convention is used throughout this section: the hexahedra are white, the prisms are yellow, the pyramids are red and the tetrahedra are

Table 2 Mesh convergence analysis (credits: Gaëtan Compère)

Mesh size	First frequency (Hz)	Second frequency (Hz)
2.0	770.11	1950.65
1.0	760.46	1928.52
0.5	757.55	1921.77
struct.	757.76	1921.93

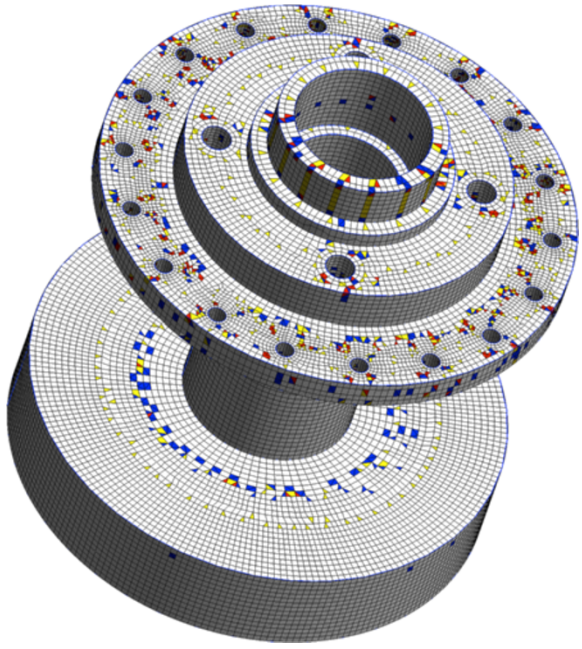

Figure 33 A motorcycle hub.

blue. The variable *NC* represents the percentage of non-conformal interior quadrilateral faces.

Figure 21 shows a mesh containing 142,466 elements. The mesh size is constant throughout the domain. The CAD model is composed of 250 geometrical faces of various sizes. In order to avoid altering the geometrical edges, hexahedra or prisms whose facets lie on two different geometrical faces are not created.

Each module described in this article shares a certain percentage of the total execution time. These percentages are detailed in Figure 22. The mesh illustrated on Figure 21 was used for the analysis. The blue modules refer to tetrahedra recombination. They are

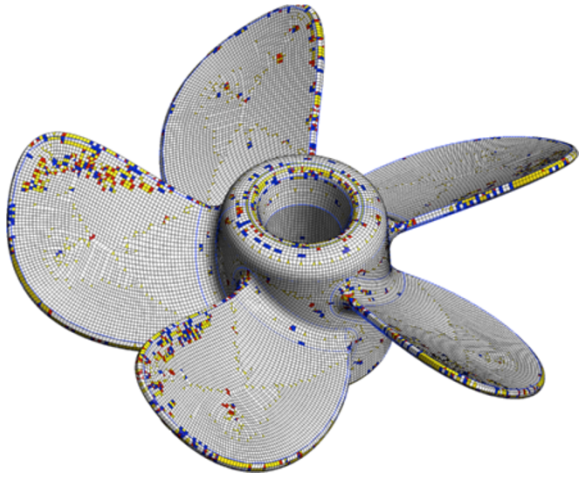

Figure 34 A marine propeller.

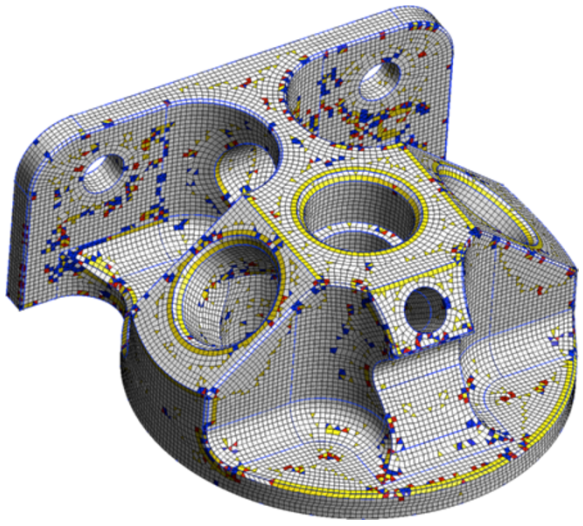

Figure 35 A filter mount.

the most time-consuming. The green and yellow modules pertain to volume and surface mesh vertices generation.

In Figure 23, a rectangular parallelepiped surrounding a NACA airfoil is meshed with 43,094 elements. Figures 24 and 25 are zoomed images of Figure 23.

Figure 26 is a cutaway view of the mixed mesh inside a cylinder. The mesh size is constant and the cardinal directions are radial.

Choosing a size field consistent with the geometry can improve the hexahedra percentage. For the spherical model shown on Figures 27 and 28, a mesh size proportional to the radius is more suitable than a constant one.

A mixed hexahedral mesh of a cylinder is displayed on Figure 29. However, this time the front was initiated only from the lower and upper disks, not from the curved face. In other words, the vertices on the curved face were not allowed to create prospective

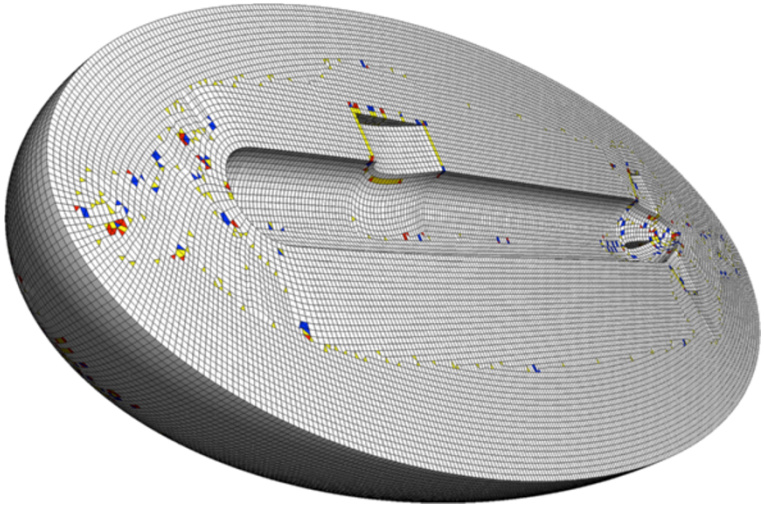

Figure 36 A submarine model.

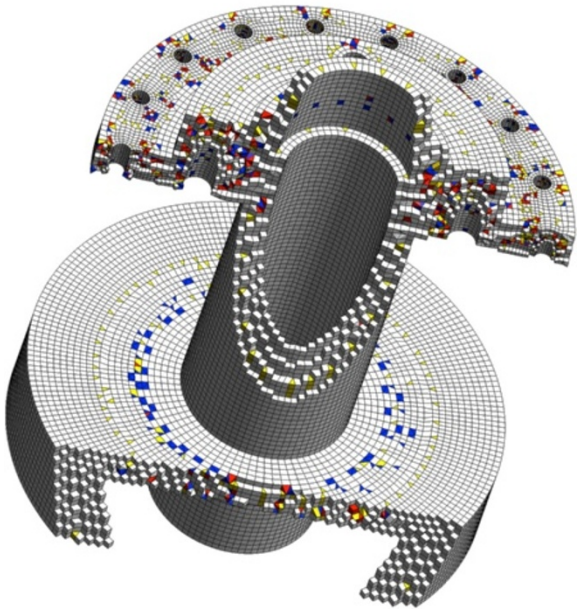

Figure 37 A cutaway view of the motorcycle hub.

vertices. The mesh has the following statistics: H_{nbr} = 81.88%, H_{vol} = 91.44% and Q = 0.97. Because of the regularity of the curved face mesh, the hexahedra percentages are much higher than those of the previous cylinder.

Figures 30 and 31 show a mixed hexahedral mesh of the anc101 mechanical part. The mesh was generated in 194 seconds and contains 92,282 elements. It has a H_{nbr} of 59.39%. The anc101 part was designed by Computer Aided Manufacturing Inc. [39] and is commonly used in mesh generation literature, in particular in Lévy and Liu's article. For approximately the same mesh size, they obtain a H_{nbr} of 77.14% and an execution time of 12 minutes. Pyramid recombination was not employed in both case.

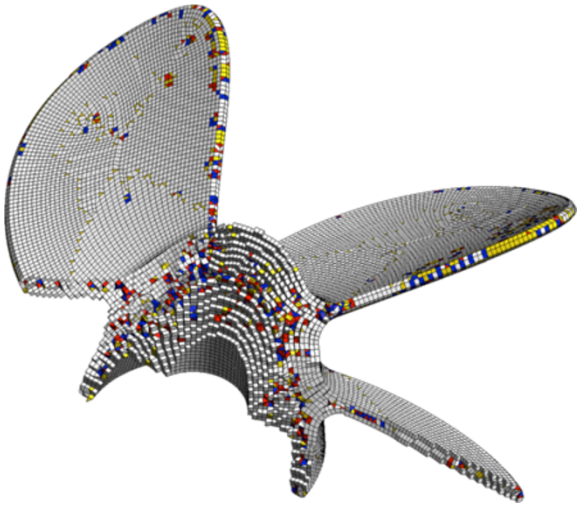

Figure 38 A cutaway view of the marine propeller.

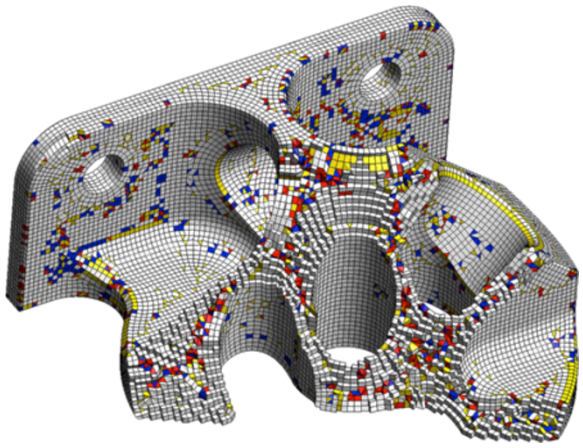

Figure 39 A cutaway view of the filter mount.

A mesh convergence analysis was performed on the mechanical piece displayed on Figure 32. The mechanical piece is made of steel and one of its extremities is fixed. The first and second frequencies are computed, as shown on Table 2. Mixed hexahedral meshes of various densities are employed. According to a similar finite element calculation performed on a structured mesh, the first frequency is equal to 757.76 Hz and the second one is equal to 1921.93 Hz. The results appear to converge.

Figures 33, 34, 35 and 36 present additional examples of mixed hexahedral meshes. Figures 37, 38, 39 and 40 show cutaway views of these meshes. Table 3 contains the corresponding mesh data. The CAD models of these meshes come from an online repository [40-42].

A frequency-domain computational acoustic simulation was performed on the submarine model SUB of Table 3, under plane wave incidence. Figure 41 shows a cutaway view

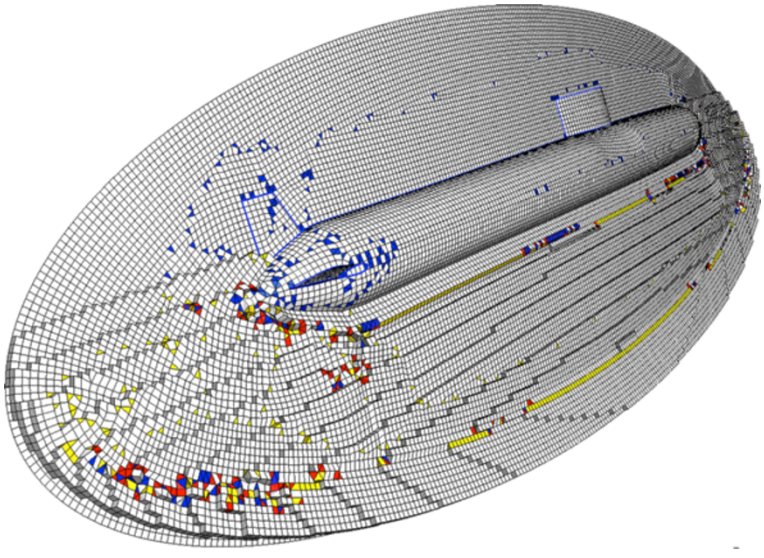

Figure 40 A cutaway view of the submarine model.

Table 3 Mesh data

Figure	# vertices	H_{vol}	*CPU (s)*
33	133,436	89.74%	247 s.
34	133,678	83.65%	268 s.
35	102,946	78.55%	225 s.
36	598,514	90.28%	1287 s.

The execution times were measured on a 2012 laptop computer.

of the iso-surfaces of the diffracted pressure field. Again, the simulations were shown to converge with mesh refinement.

Surfaces and volumes are meshed sequentially. The two surface meshes bounding a thin region may also not be identical. As a consequence, many non-hexahedral elements can be created. The algorithm is usually less effective for geometrical models featuring many thin regions.

Conclusion

A method capable of generating mixed hexahedral meshes has been presented. The first step consists of covering geometrical boundaries with aligned vertices using a frontal process. The interior is treated in a similar fashion. Vertices creation are guided by a direction field and a size field. The interior vertices are eventually tetrahedralized with a Delaunay procedure. All tetrahedra combinations yielding hexahedra are identified. They are sorted by quality and the highest quality hexahedra are created first. The same approach is applied to prisms. The final mesh contains hexahedra, prisms, pyramids and remaining tetrahedra.

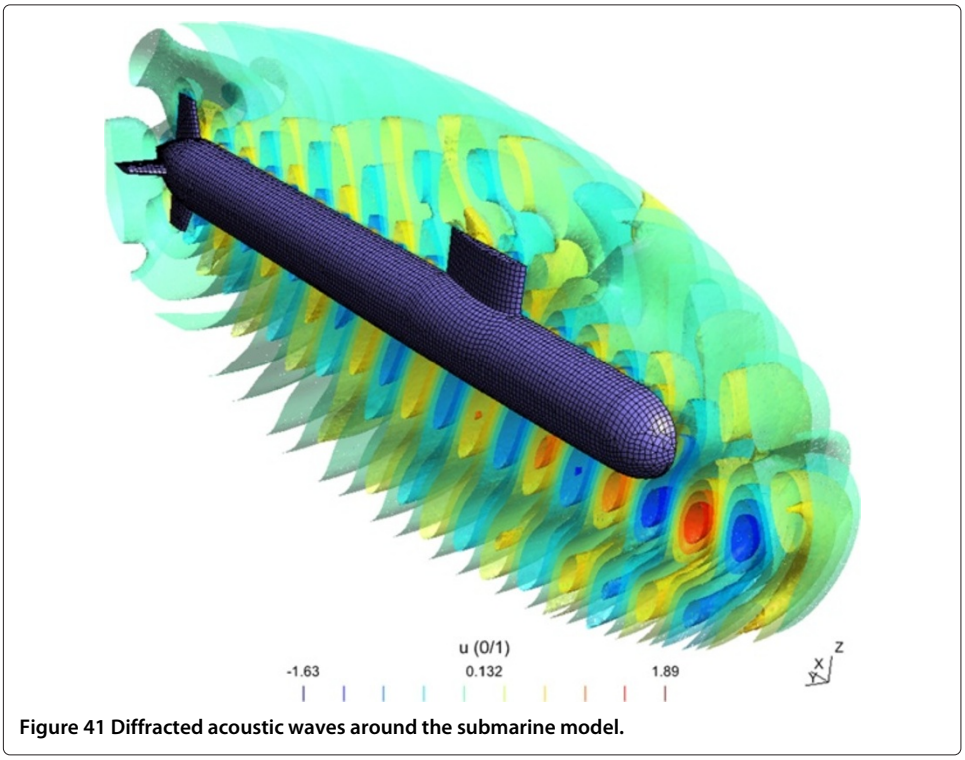

Figure 41 Diffracted acoustic waves around the submarine model.

The method has obvious drawbacks. First, there are no guarantees regarding the hexahedra percentage. It can be higher for certain geometries and lower for others. Secondly, the hexahedra are not anisotropic. For many geometries, well chosen anisotropy could increase the number of hexahedra. Finally, the resulting meshes are useful only to solvers capable of handling a certain number of non-conformal faces.

Competing interests
The authors declare that they have no competing interests.

Authors' contributions
TCB worked on the algorithms and drafted some parts of the manuscript. JFR, EM and FH worked on the algorithms, drafted some parts of the manuscript and carried out detailed revisions. CG performed acoustic finite element analyses with hex-dominant meshes and carried out detailed revisions of the manuscript. All authors read and approved the final manuscript.

Acknowledgements
This work has been partially supported by the Belgian Walloon Region under WIST grants ONELAB 1017086 and DOMHEX 1017074. The authors wish to thank Gaëtan Compère for the mesh convergence analysis and Jonathan Lambrechts for suggesting the use of R-trees. The authors also appreciate the reviewers' efforts and suggestions.

Author details
[1]Université catholique de Louvain, Institute of Mechanics, Materials and Civil Engineering, Bâtiment Euler, Avenue Georges Lemaître 4, Louvain-la-Neuve 1348, Belgium. [2]Université de Liège, Dept. of Electrical Engineering and Computer Science, Montefiore Institute, Bâtiment B28, Sart-Tilman, Liège 4000, Belgium.

Received: 19 August 2013 Accepted: 16 January 2014
Published: 10 February 2014

References

1. Benzley SE, Perry E, Merkley K, Clark B, Sjaardema G (1995) A comparison of all hexagonal and all tetrahedral finite element meshes for elastic and elasto-plastic analysis. In: Tautges T (eds) Proceedings of the 4th International Meshing Roundtable. Sandia National Laboratories, Albuquerque
2. Puso MA, Solberg J (2006) A stabilized nodally integrated tetrahedral. Int J Numer Meth Eng 67:841–867
3. Ito Y, Nakahashi K (2004) Improvements in the reliability and quality of unstructured hybrid mesh generation. Int J Numer Meth Fl 45:79–108
4. Pirzadeh S (1996) Three-dimensional unstructured viscous grids by the advancing-layers method. AIAA J 34:43–49
5. Kallinderis Y, Ward S (1993) Prismatic grid generation for three-dimensional complex geometries. AIAA J 31(10):1850–1856
6. Shepherd JF, Johnson CR (2008) Hexahedral Mesh Generation Constraints. Eng Comput 24:195–213. https://dl.acm.org/citation.cfm?id=1394234.
7. Meyers RJ, Tautges TJ, Tuchinsky PM (1998) The 'Hex-Tet'hex-dominant meshing algorithm as implemented in CUBIT. In: Freitag L (ed) Proceedings of the 7th international meshing roundtable. Sandia National Laboratories, Dearborn, pp 151–158
8. Dewhirst DL, Grinsell PM, Tucker JR, Mahajan A (1993) Joining tetrahedra to hexahedra. In: Proceedings of MSC World Users' Conference. MSC Software, Arlington. http://web.mscsoftware.com/support/library/conf/wuc93/p04593.pdf.
9. Yamakawa S, Shimada K (2003) Fully-automated hex-dominant mesh generation with directionality control via packing rectangular solid cells. Int J Numer Meth Eng 57:2099–2129
10. Shewchuk JR (1998) Tetrahedral mesh generation by Delaunay refinement. In: Proceedings of the fourteenth annual symposium on Computational geometry. ACM, Minneapolis, pp 86–95
11. Geuzaine C, Remacle JF (2009) Gmsh: a three-dimensional finite element mesh generator with built-in pre- and post-processing facilities. Int J Numer Meth Eng 79:1309–1331
12. Schneiders R, Schindler R, Weiler F (1996) Octree-based Generation of Hexahedral Element Meshes. In: Proceedings of the 5th International Meshing Roundtable. Sandia National Laboratories, Pittsburgh
13. Ito Y, Shih AM, Soni BK (2008) Octree-based reasonable-quality hexahedral mesh generation using a new set of refinement templates. Int J Numer Meth Eng 77:1809–1833
14. Maréchal L (2009) Advances in Octree-Based All-Hexahedral Mesh Generation: Handling Sharp Features. In: Clark BW (ed) Proceedings of the 18th International Meshing Roundtable. Springer, Salt Lake City
15. Blacker TD (2000) Meeting the Challenge for Automated Conformal Hexahedral Meshing. In: Owen S (ed) Proceedings of the 9th International Meshing Roundtable. Sandia National Laboratories, New Orleans
16. Huang J, Tong Y, Wei H, Bao H (2011) Boundary aligned smooth 3D cross-frame field. In: Bala K (ed) Proceedings of ACM SIGGRAPH Asia. Association for Computing Machinery, Hong Kong, p 2011
17. Li Y, Liu Y, Xu W, Wang W, Guo B (2012) All-hex meshing using singularity-restricted field. In: Sloan P (ed) Proceedings of ACM SIGGRAPH Asia. Association for Computing Machinery, Singapore, p 2012
18. Huang J, Jiang T, Wang Y, Tong Y, Bao H (2012) Automatic Frame Field Guided Hexahedral Mesh Generation. Tech. rep., Zhejiang University. http://www.cad.zju.edu.cn/home/hj/12/hex/techreport/hex-techreport.pdf.
19. Nieser M, Reitebuch U, Polthier K (2011) CubeCover - parameterization of 3D volumes. Comput Graph Forum 30:1397–1406
20. Kowalski N, Ledoux F, Frey P (2012) A PDE Based Approach to, Multidomain Partitioning and Quadrilateral Meshing. In: Jiao X, Weill J (ed) Proceedings of the 21th International Meshing Roundtable. Springer, San Jose

21. Gregson J, Sheffer A, Zhang E (2011) All-Hex mesh generation via volumetric polyCube deformation. Comput Graph Forum 30:1407–1416
22. Meshkat S, Talmor D (2000) Generating a mixed mesh of hexahedra, pentahedra and tetrahedra from an underlying tetrahedral mesh. Int J Numer Meth Eng 49:17–30
23. Du Q, Faber V, Gunzburger M (1999) Centroidal Voronoi Tessellations: Applications and Algorithms. Siam Rev 41:637–676
24. Lévy B, Liu Y (2010) L_p Centroidal Voronoi Tessellation and its Applications. In: Hoppe H (ed) Proceedings of ACM SIGGRAPH 2010. Association for Computing Machinery, Los Angeles
25. Ray N, Vallet B, Li WC, Lévy B (2008) N-Symmetry direction field design. ACM T Graph 27:1–25
26. Guttman A (1984) R-Trees: A Dynamic Index Structure for Spatial Searching. In: ACM Special Interest Group on Management of Data. Association for Computing Machinery, Boston, pp 47–57
27. Douglas G, Green M, Guttman A, Stonebraker M (2004) R-trees: a dynamic index structure for spatial searching. http://www.superliminal.com/sources/RTreeTemplate.zip.
28. Vyas V, Shimada K (2009) Tensor-Guided Hex-Dominant Mesh Generation with Targeted All-Hex Regions. In: Clark BW (ed) Proceedings of the 18th International Meshing Roundtable. Springer, Salt Lake City
29. Remacle JF, Geuzaine C, Compère G, Marchandise E (2010) High quality surface meshing using harmonic maps. Int J Numer Meth Eng 83:403–425
30. Marchandise E, de Wiart CC, Vos WG, Geuzaine C, Remacle JF (2011) High-quality surface remeshing using harmonic maps-Part II: Surfaces with high genus and of large aspect ratio. Int J Numer Meth Eng 86:1303–1321
31. Marchandise E (2013) Remacle JF. Eng Comput. doi:10.1007/s00366-012-0309-3
32. Remacle JF, Henrotte F, Baudouin TC, Geuzaine C, Béchet E, Mouton T, Marchandise E (2011) A Frontal Delaunay quad mesh generator using the L^∞ norm. In: Quadros W (ed) Proceedings of the 20th International Meshing Roundtable. Springer, Paris
33. Mount DM, Arya S (1997) ANN: A library for approximate nearest neighbor searching. In: CGC Workshop on Computational Geometry. The Center for Geometric Computing, Durham, pp 33–40
34. Remacle JF, Lambrechts J, Seny B, Marchandise E, Johnen A, Geuzaine C (2012) Blossom-Quad: a non-uniform quadrilateral mesh generator using a minimum cost perfect matching algorithm. Int J Numer Meth Eng 89:1102–1119
35. Si H (2006) TetGen: A Quality Tetrahedral Mesh Generator and Three-Dimensional Delaunay Triangulator. http://tetgen.berlios.de/files/tetgen-manual.pdf
36. Baudouin TC, Remacle JF, Marchandise E, Lambrechts J, Henrotte F (2012) Lloyd's energy minimization in the L_p norm for quadrilateral mesh generation. Eng Comput. DOI:10.1007/s00366-012-0290-x
37. Fortin A, Garon A (2014) Les Eléments finis : de la théorie à la pratique. http://giref.ulaval.ca/files/afortin/Publications/elements_finis.pdf.
38. Owen SJ, Canann SA, Saigal S (1997) Pyramid Elements for Maintaining Tetrahedra to Hexahedra Conformability. In: Canann S, Saigal S (ed) ASME Trends in Unstructured Mesh Generation. American Society of Mechanical Engineers, Evanston, pp 123–129
39. Agoston MK (2005) Computer Graphics and Geometric Modeling. Springer-Verlag, USA
40. Rieling R (2013) Yamaha XTZ-125. http://grabcad.com/library/yamaha-xtz-125.
41. 5 Bladed Propeller (2011). http://grabcad.com/library/5-bladed-propeller.
42. Hall C (2013) CV HP1. http://grabcad.com/library/cv-hp1.

doi:10.1186/2213-7467-1-8
Cite this article as: Carrier Baudouin *et al.*: **A frontal approach to hex-dominant mesh generation.** *Advanced Modeling and Simulation in Engineering Sciences* 2014 **1**:8.