Frame field smoothness-based approach for hex-dominant meshing

P.-E. Bernard[a,*], J.-F. Remacle[a], N. Kowalski[a], C. Geuzaine[b]

[a] *Université catholique de Louvain, Institute of Mechanics, Materials and Civil Engineering (iMMC), Bâtiment Euler, Avenue Georges Lemaître 4, 1348 Louvain-la-Neuve, Belgium*
[b] *Université de Liège, Department of Electrical Engineering and Computer Science, Montefiore Institute B28, Grande Traverse 10, 4000 Liège, Belgium*

Abstract

An indirect approach for building hex-dominant meshes is proposed: a tetrahedral mesh is constructed at first and is recombined to create a maximum amount of hexahedra. The efficiency of the recombination process is known to significantly depend on the quality of the sampling of the vertices. A good vertex sampling depends itself on the quality of the underlying frame field that has been used to locate the vertices. An iterative procedure to obtain a high quality three-dimensional frame field is presented. Then, a new point insertion algorithm based on a frame field smoothness is developped. Points are inserted in priority in smooth frame field regions. The new approach is tested and compared with simpler strategies on various geometries. The new method leads to hex-dominant meshes exhibiting either an equivalent or a larger volume ratio of hexahedra (up to 20%).

Keywords: hexahedral meshing ; mixed hexahedral meshes ; tetrahedra recombination

1. Introduction

Hexahedral meshes are commonly preferred to tetrahedral meshes in engineering analysis. One of their main advantage resides in the fact that a lower number of elements is required for the same amount of vertices, compared to

*paul-emile.bernard@uclouvain.be

tetrahedra. In computational fluid dynamics, hexahedral meshes offer for instance good results along boundary layers. Indeed, the anisotropic refinement of tetrahedra is known to produce poor quality elements [1], while this operation does not affect the quality of hexahedra. In the field of computation solid mechanics, tetrahedra may also lead to some issues as inaccuracy or locking problems [2]. Understanding precisely why an element type is better than another in some situations may be still open for debate. It is indeed a fact is that, even though the generation of tetrahedral meshes can be considered as a well known problem, the automatic generation of conforming all-hexahedra meshes is still an open issue.

Mesh generation has always been considered as a time-consuming process in engineering analysis. The generation of tetrahedral meshes may now be considered merely automatic. The generation of all-hex and hex-dominant meshes often requires time consuming user interactions. Our purpose here is to develop a fully automated procedure for generating non uniform hex-dominant meshes that contain a maximum amount of hexahedra, both in volume and number.

The main idea of the current approach is to decompose the meshing procedure, as proposed in [3], in two different stages: first, bulk points are created inside the domain and are subsequently tetrahedralized; then, tetrahedra are recombined to create a mixed mesh containing a maximum amount of quality hexahedra. The way points are distributed in the domain is of paramount importance for obtaining a mesh that is truly hex-dominant. The point insertion algorithm that we have proposed in [3] was based on a first in first out approach: points of the boundary of the domain were inserted in a queue and each of those was generating 6 potential neighbors along pre-defined directions. Each potential point was then inserted in the queue if it is not too close to an existing point.

The approach of [3] allowed us to generate high quality hex-dominant meshes with sometimes more than 90% of hexes in volume. Yet, this simple fifo approach

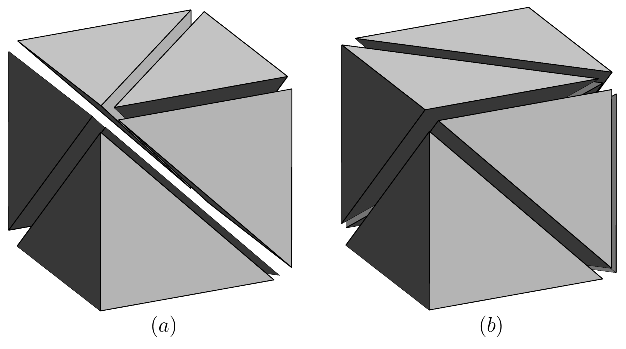
Figure 1: Two examples of recombination patterns.

is clearly perfectible. This is the object of this paper.

In this work, we start by developing a smoothness indicator that allow to identify the geometric singularities of the domain as well as regions where the geometry is smooth. A new point insertion procedure is then proposed in which the points are prioritarly inserted in smooth regions where the probability of generating large chunks of structured hexes is high.

The point insertion method and the construction of the underlying frame field is developed in section 3. In section 2, we present the algorithm for recombining tetrahedra into hexahedra. Some results are presented in section 4.

2. Recombination: from tetrahedra to hexahedra

2.1. Recombination patterns

One single hexahedron can be decomposed into 5, 6, or 7 tetrahedra [4]. Figure 1 shows 2 of the possible patterns. The problem of finding all the possible recombination patterns in a tetrahedral mesh may seem intractable but it is not. Two hexes that do not share any vertex cannot be recombined into one hexahedron. Thus, searching all potential hexahedra in a tet mesh is the sum of local problems: each tetrahedron T of the mesh is associated to a cavity

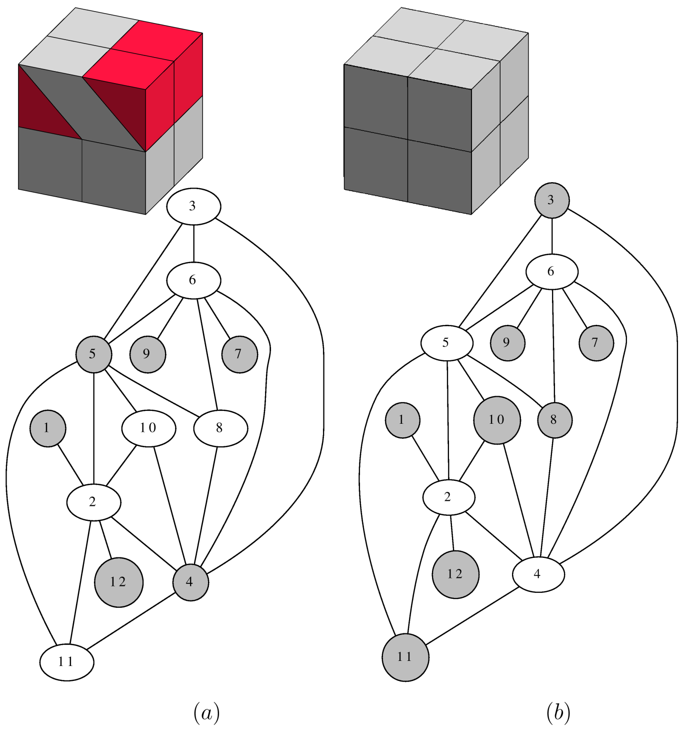

Figure 2: The maximum clique, on the right, contains 8 hexahedra, while another clique of 6 hexahedra yields the mesh on the left, with 4 prisms in red. The graphs depicted here are incompatibility graphs: the independent sets corresponding to the meshes are highlighted in gray.

that contains all tetrahedra that share vertices with T. Then, patterns are searched into that small subset of the mesh. This procedure yields an extensive set of potential hexahedra. If two of the potential hexahedra share one ore more tetrahedra, they are obviously mutually exclusive. More subtle incompatibilities have to be taken into account like hexahedra that would share 3 points on one of their face or that would share one face diagonal.

2.2. Potential hexahedra as a graph

Consider the set of all potential hexahedra $\mathcal{H} = \{H_1, \dots, H_{N_h}\}$. Here, N_h is typically of the order of N_t if N_t is the number of tetrahedra in the mesh (see Table 1). We build the following undirected graph G : each node i of G

correspond to a potential hexahedron and edge ij exists if H_i and H_j are compatible i.e. if those two hexahedra can exist simultaneously in a finite element mesh. A subset $S = \{H_{i_1}, \dots, H_{i_m}\}$ of m hexahedra that are all compatible with each other form a clique i.e. a subgraph of G such that any pair of nodes are connected nodes. To maximize the number of hexahedra in the final mesh, we thus need to find the largest clique possible.

In this specific problem, we must remind that two potential hexahedra that share no vertices are always compatible. This implies that, asymptotically, nearly all hexahedra are compatible with each other which means that the number of edges in G is close to $N_h^{N_h}$, which is huge (a very standard mesh can have $N_h = 10^6$ hexahedra). Consider now the the dual of G i.e. a graph G' with the same nodes but where an edge exists between H_i and H_j if and only if it was not present in G. This graph connects hexahedra that are incompatible. Compatible elements being clustered, the number of edges in G' is of order $\mathcal{O}(N_h)$. Note that finding the maximal clique of G is equivalent to finding the maximal independent set of the dual graph G'.

The maximal independent set can be defined as the larger subgraph S, i.e. the independent set with the highest m. Weights w_j can also be defined on the nodes of the graph and the maximum independent set can be defined as the one that maximizes $\sum_{j=1}^{m} w_j$.

Unfortunately, the general problem of the maximum clique/independent set is known to be NP-hard. The complexity for finding all the maximum cliques, in the worst-case, varies like $\mathcal{O}(\alpha^n)$ for n nodes.

In the algorithm proposed in [5], all the maximum cliques are found. At some point, the decision criteria to choose a node is a function f, equal to n_c, the number of compatible adjacent nodes. Of course, one could change this function to be maximized and make it depend on other criteria. For instance, one could choose a weighted sum including the element quality and the boundary proximity to create this function to maximize.

On Figure 2, the algorithm from [5] has been used to find the maximum clique (depicted on Figure 2(b)) on a very simple cubic domain. The optimal solution made of 8 hexahedra is found. The corresponding incompatibility graphs of potential hexahedra are shown, with highlighted independent sets in gray on Figure 2. We observe that the number of potential hexahedra in the graph is $N_h = 12$ and the average degree of a node (average number of connections in the graph G') is $C(G') \approx 3$. In Table 1 are summarized some statistics for larger graphs on the cube and the dome geometry from Figure 8(a).

Table 1: Number of potential hexahedra N_h, of initial tetrahedra N_t, and average degree C in graph G'.

Geometry	N_t	N_h	$C(G')$
Cube $(3 \times 3 \times 3)$	162	181	68
Cube $(4 \times 4 \times 4)$	387	469	76
Cube $(5 \times 5 \times 5)$	756	992	86
Dome (Fig.8(a))	5757	5390	77

However, since such a clique algorithm presents a high computational cost, we use hereafter a simple *greedy algorithm* for meshes involving a large amount of elements, while keeping the idea of the function to maximize. The greedy algorithm consists in sorting all the potential hexahedra, according to this weighted function, and choosing in priority the ones with the higher values. Note that if the optimal solution from the clique problem is obviously the optimal solution for the greedy algorithm in the 8 hexahedra cube, this is of course not a general result.

Let us finally mention that, depending on the kind of graph involved in this mesh generation problem, one could consider using appropriate heuristics for these graphs. Such heuristics could allow one to use faster algorithms for solv-

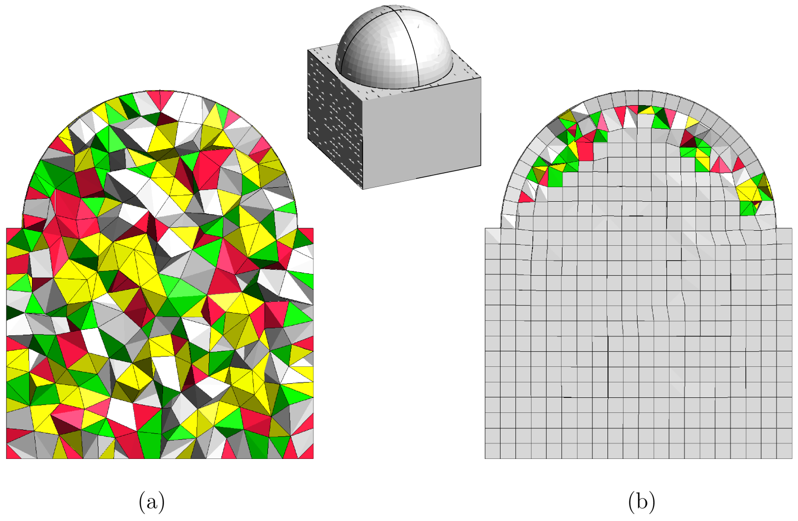
Figure 3: Cut view of a dome meshed using standard Delaunay point insertion (a) and the new frame field based approach (b), presenting a volume ratio of hexahedra of 38.6% and 94.8% respectively. Hexahedra are colored in gray, while the colors yellow, red and green correspond to tetrahedra, prisms and pyramids respectively.

ing the maximum clique (or approximate maximum) problem.

3. Point insertion

In two dimensions, we have shown in [6] that any mesh composed of an even number of triangles could be converted into a quad mesh using edge swaps and recombination. The quality of the final quand mesh benefits of course of an appropriate point insertion scheme [7] but there is a guaranty of finding a mesh composed of quadrangles only. In 3D, things are very different. The recombination algorithm that we have proposed here-above does not guarantee that every tetrahedra will be converted into an hexahedra. In 3D, the percentage of hexahedra in the final mesh strongly depends on the point insertion scheme. This sensitivity is illustrated on Figure 3 where a dome is meshed using a standard Delaunay refinement approach and using our new approach. Here, we do not

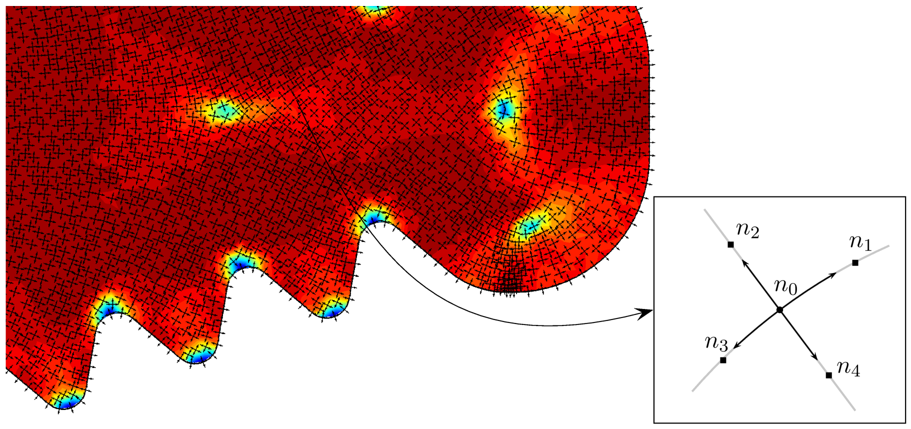

Figure 4: Close up view of the 2D frame field of the hollow turbine blade depicted on Figures 8(c) and 11, the color field representing its smoothness. The frame field directions around point n_0 are represented with gray curves (on the right). For such a smooth frame field, four (or six in 3D) additional points $n_i,\ i = 1 \ldots 4$, are correctly computed. On the contrary, geometric singularities located in the blue spots may lead to inaccurate additional points.

consider any post-smoothing of the points positions (as, for instance, in [8, 9]) to improve the mesh quality, but we propose a pre-computation to directly obtain appropriate point locations. This *a priori* approach is based on two main ingredients: a frame field that will guide point insertion and the point insertion scheme itself i.e. the way points are prioritized in the insertion queue.

3.1. The two-dimensional frame field

Let us briefly recall the main idea for computing the two-dimensional frame field.

A frame field is a set of two orthonormal tangent vector fields that vary smoothly on the $2D$ manifold. Frame fields may be forced to aligned with the boundary of the domain or to the principal directions of curvature of a surface. Frame fields are mathematical objects that have nasty symmetries: in 2D, any rotation of $k\pi/4,\ k \in Z$ makes the frame field invariant. In 3D, there are 24 symmetries.

In 2D, computing frame fields is straightforward, using e.g. finite elements. Let θ be the local angle between the frame field and a reference basis at a given point. The frame field at any point of the surface is found by solving an elliptic PDE on the surface Ω with boundary conditions that enforce θ on the domain boundary $\partial\Omega$:

$$\begin{aligned} \nabla^2 a(\theta) = 0 \quad \text{on } \Omega, \qquad & a(\theta) = \bar{a} \quad \text{on } \partial\Omega \\ \nabla^2 b(\theta) = 0 \quad \text{on } \Omega, \qquad & b(\theta) = \bar{b} \quad \text{on } \partial\Omega \end{aligned} \tag{1}$$

with $a(\theta) = \cos(4\theta)$, $b(\theta) = \sin(4\theta)$, and $\bar{a}, \bar{b}$ the boundary conditions set in such a way that the frame field is aligned on the outgoing normal vector to the domain. The frame field is eventually given by

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

Details about frame field computation can be found in [7, 3]. On Figure 4 is depicted an example of 2D frame field.

3.2. Point insertion

The aime of a point insertion algorithm is to build a set of points in the domain that are aligned with the frame field and that have a density that is predetermined. Our approach is based on a priority queue. All the points of the boundary of the domain are initially inserted in the queue. Then, the point at the tail of the queue is removed from the queue and 6 potential points in 3D or 4 potential points points in 2D are created along the local directions of the frame field (see Figure 4). Those potential points are inserted in the queue provided that they are not too close to existing points of the set of points and that they lie in the domain. In [3], the priority queue was a *fifo* queue i.e. the priority was “first in, first out”. This queueing system implies that new points are inserted layer by layer, starting from the boundaries and moving towards the inside of the domain. This first approach looks very much like a frontal approach: fronts are colliding at some point, typically close to the medial axis of the domain. This behavior is visible on the left image of Figure 5. On this same picture, it is clear that in those regions where the frame field is smooth,

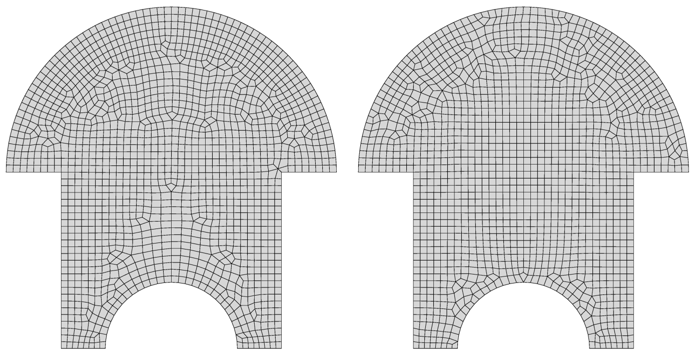

Figure 5: Quad meshes obtained using the *fifo* frontal algorithm (left) and the smoothness-based algorithm (right).

the point insertion scheme produces very good results. On the other hand, when fronts are reaching non smooth regions or are close to singularities, it becomes less easy to produce optimal point locations. Figure 4 shows frame field smoothness for that particular domain. Singularities are localized in the blue spots: we observe large variation of the frame field around these points. The idea here is to change the way the queue is prioritized: points should first be inserted where the frame field is smoother. Points close to singularities will be inserted at the end, leaving the algorithm fill up large parts of the domain bulk where frame fields are smooth.

Two issues must be addressed at this point. First, the definition of smoothness itself. Then, as we will observe in Section 4, the fact that a good point insertion algorithm may be quite useless if based on a poor 3D frame field: it must be smoothed as well.

3.3. Building the three-dimensional frame field

Similarly to the two-dimensional meshing procedure, we start with a tetrahedral background mesh to extend the frame field and mesh size field from the boundaries to the volume. But unlike the two-dimensional case, describing a frame requires three degrees of freedom instead of one. It is indeed not possible

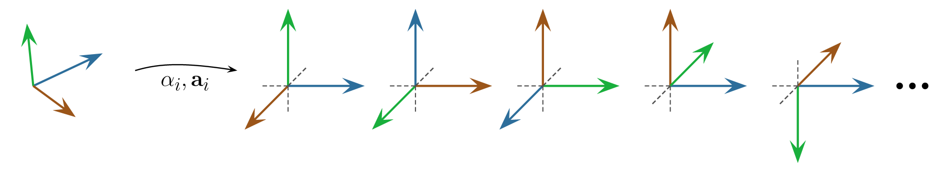

Figure 6: The rotations applying a frame on another are each characterized by an angle $\alpha_i \geq 0$ and an axis $\boldsymbol{a}_i$, $i = 1 \dots 24$.

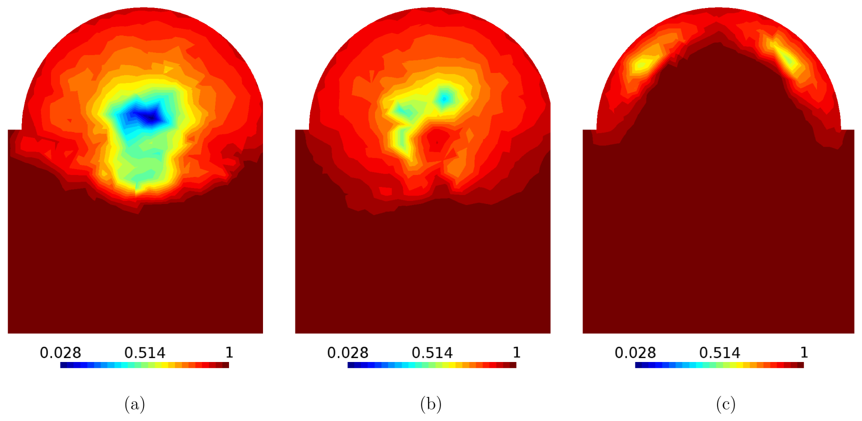

Figure 7: Cut views of the frame field smoothness for the dome: using the nearest neighbor on boundary (a), smoothing with constant diffusivity (b) and a variable diffusivity (c).

to formulate the smoothing of a 3D frame field as an elliptic PDE problem. Here, we choose to consider a local smoothing method where a single frame is aligned with its mesh-neighbors.

First, let us consider only two frames. There are 24 possible rotations to apply a frame on another (some of these are depicted on Figure 6), each rotation being characterized by an angle $\alpha_i \geq 0$ and an axis $\boldsymbol{a}_i$. We compute these rotations, and choose to consider the one with the smaller angle α, the other transformations are discarded.

Then, let us consider a node n_0 surrounded by N neighbors n_i, $i = 1 \dots N$. We have chosen N rotations applying the frame of n_0 on the frame of each n_i, i.e. N couples $(\boldsymbol{a}_i, \alpha_i)$. Smoothing is done by applying the following rotation of axis $\boldsymbol{a}$ and angle α to the frame of n_0:

$$\boldsymbol{a}^* = \sum_{i=1}^{N} \boldsymbol{a}_i \alpha_i \ , \qquad \alpha = ||\boldsymbol{a}^*|| \ , \qquad \boldsymbol{a} = \frac{\boldsymbol{a}^*}{\alpha} \ . \tag{2}$$

Note that for $N \geq 2$, several iterations may be required to obtain convergence of the frame of n_0. In the computations of Section 4, we considered that convergence is reached when the angular difference $\Delta\alpha < 5\ 10^{-3}$radians $\approx 0.29°$. In practice, when the whole global frame field is not converged, we observed an average of 2.1 local iterations, while this number gets closer to 1 as the global frame field converges.

We apply this local transformation to the frame field at each node of the 3D background mesh. Then, we compute the infinity norm $N_\infty = \max(\alpha)$ as the maximum angle of rotation used to align a frame on its neighbors, and iterate until convergence of this norm. In the computations of Section 4, we considered that convergence is reached when $N_\infty < 3\ 10^{-2}$ radians, i.e. when the larger rotation angle encountered in the whole domain is smaller than about 1.7°.

These iterations eventually correspond to some pseudo-diffusion operator. We imposed a Dirichlet boundary condition, which is the 2D frame field computed using the PDE (1). For the initial condition of the iterative smoothing, the frame field is given the value of its nearest frame on the boundary. We observed a huge importance of that initial condition. Indeed, using for instance a totally random frame field as initial condition leads to a totally different converged field. This iterative smoothing does not erase geometric singularities and does not create new ones either: this is why it is mandatory to initiate the frame field with a good first guess.

Nodes are sorted with respect to their angular difference between two iterations. This is done to improve the convergence. The less converged nodes are treated last like in a Gauss-Seidel method. Finally, computational costs are strongly reduced by stating that if the frames of a node and its neighbors were not modified during the previous iteration (i.e. the criterion $\Delta\alpha$ mentioned above was directly satisfied), computation is unnecessary for that node.

While the frame field is smoothed, we also evaluate its smoothness s at each node, simply defined as

$$s = 1 - \frac{4}{\pi N}\sum_{i=1}^{N} \alpha_i . \tag{3}$$

The smoothness is thus simply a measure of the average angle α to apply the frame of node n_0 on each neighbor, with a scaling to obtain a lower bound of approximately 0 and a maximum value of 1 for perfectly aligned frames.

Figure 7 depicts the frame field smoothness in the center of the dome: in Figure 7(a), the frame field is not smoothed, but is equal to the frame field of the nearest neighbor on the boundary, while in Figure 7(b) is depicted the result of the iterative procedure described above. However, the result in 7(b) does not seem sufficient to really improve the ratio of hexahedra. We also would like to reduce the impact of the singularities on their direct neighborhood, and therefore extend the regions presenting a smooth frame field.
To achieve this, we slightly modify the rotation (2) and introduce different weights c_i depending on the local smoothness s_i:

$$\boldsymbol{a}^* = \sum_{i=1}^{N} c_i(s_i)\boldsymbol{a}_i\alpha_i \ , \qquad \alpha = ||\boldsymbol{a}^*|| \ , \qquad \boldsymbol{a} = \frac{\boldsymbol{a}^*}{\alpha} \ , \tag{4}$$

with a relationship as for instance

$$c(s) = \begin{cases} 1 & \text{if} \quad s \geq 0.85 \\ 10^{-3} & \text{if} \quad s < 0.85 \end{cases} \tag{5}$$

The iterative smoothing procedure then becomes similar to a diffusion operator with a variable diffusivity, depending on local smoothness. Simply put, information strongly propagates from smooth to non smooth regions, but does not propagate much in the opposite direction.

On Figure 7(c) is depicted the resulting frame field smoothness: the impact of the singularities on the frame field is clearly diminished, leading to wide regions filled with smooth cross field.

4. Numerical results

This section is devoted to comparing the point insertion algorithms and the frame field computations described on Section 3 on the geometries depicted in Figure 8.
We will compare the final meshes, in particular the ratio of hexahedra in number and volume, as the computational time required by the different approaches. On Tables 2 and 3 are summarized the results for every geometry and method: we compare the two point insertion algorithms (frontal queue and smoothness-based ordered set) and the two frame fields (smoothed or simply based on its nearest boundary neighbor), which leads to four computations on each geometry.
As in previous figures, hexahedra are colored in gray while the colors yellow, red and green correspond to tetrahedra, prisms and pyramids respectively. As previously mentioned, when the frame field is build using the smoothing algorithm, the threshold relationship (5) is used and the frame field is initiated to its nearest neighbor value on the boundary.

The first two geometries are the dome (Figure 8 (a)) and the Stanford bunny (Figure 8 (b)). On Figures 9 and 10 are depicted cut views of the 3D mesh for the frontal approach and the frame field-based approach. We observe a large improvement using the new approach: the half-sphere of the dome is filled with hexahedra, while the bunny presents large blocs of aligned hexahedra.
This is confirmed by the hexahedra ratios in Table 2. The new cross-field based approach is particularly efficient on these two geometries for the same reason: they prensent a large volume for a small amount of boundaries. However, the difference between the two is that the faces of the dome are straight while the

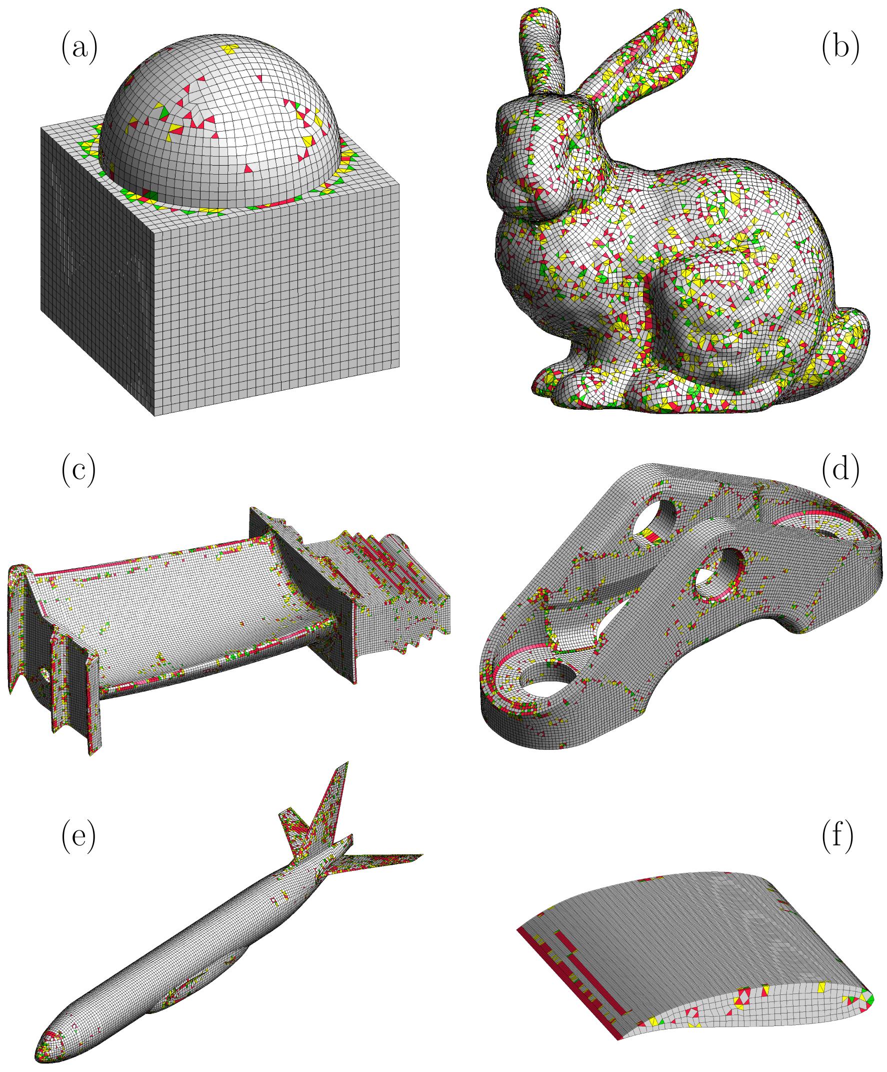

Figure 8: The six geometries used to compare the different approaches.

Table 2: The times t_b, t_i and t_r are the times (on a single laptop computer) for background mesh and smoothing operations, for point insertion and for tetrahedra recombination respectively, t_t is the total time for meshing. For each test case, the frame field can be computed using the nearest neighbor (NN) or the smoothing iterative procedure (Sm), and the insertion algorithm can be either frontal using a queue (Fr) or frame field-based using an ordered set (FB).

Test case	Nb. elements	Time (s)				Hex ratio (%)	
		t_b	t_i	t_r	t_t	Nb	Vol
Dome (Fig.9)							
(NN-Fr)	34864	1.2	4.8	30.1	39.3	62.1	89.7
(NN-FB)	33741	1.2	4.1	24.8	32.9	64.9	90.9
(Sm-Fr)	32623	6.3	3.5	23	35.6	68.8	92.2
(Sm-FB)	27115	6.3	4.4	20.1	33.8	84	96.6
Bunny (Fig.10)							
(NN-Fr)	252246	20.9	27.9	204.7	272.5	23.7	60.7
(NN-FB)	218843	21.5	35.3	178.4	250.8	30.1	68.4
(Sm-Fr)	226083	152.5	24.1	177.4	367.4	29.8	66.9
(Sm-FB)	175541	150.6	29.1	133.8	327.2	43.2	79.1
Blade (Fig.11)							
(NN-Fr)	177075	86.6	72.4	138.5	318	45.3	80
(NN-FB)	169525	105	90.8	135.6	356	48.3	81.4
(Sm-Fr)	168916	132.5	68	124	344.7	48.5	81.7
(Sm-FB)	155691	111.5	61	115.5	308.4	55.4	85
Piece (Fig.8(d))							
(NN-Fr)	99080	29	27.3	71.1	142.4	55.1	85.2
(NN-FB)	98546	27.8	43.8	82.5	170	55.8	85.5
(Sm-Fr)	96783	50.1	20.3	73.7	157.3	57.3	86.1
(Sm-FB)	96458	61.9	27.7	69.5	172.6	58.2	86.6

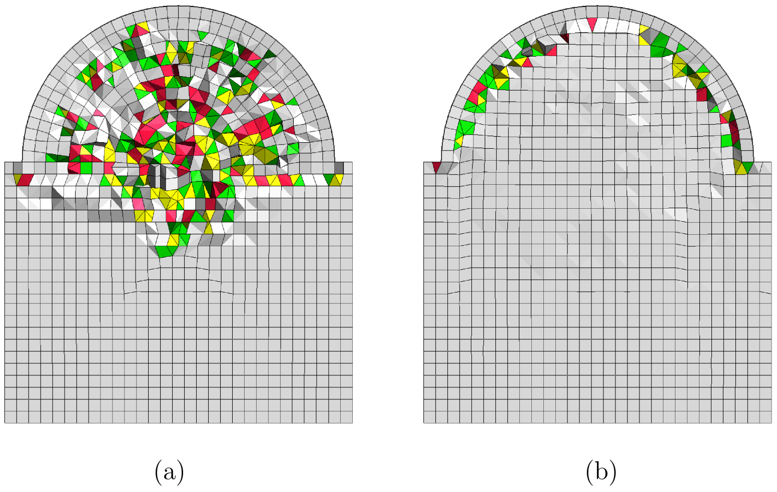

Figure 9: Cut view of the mesh, in the center of the dome: frontal (a) and smooth frame field (b) approaches.

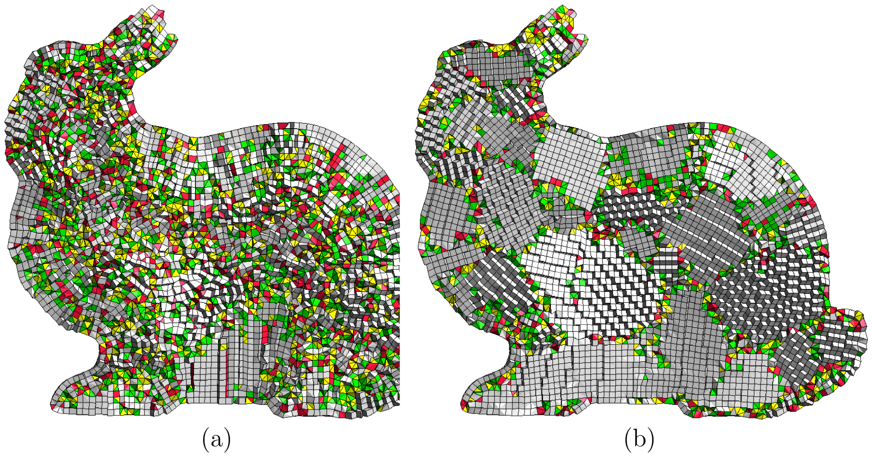

Figure 10: Cut view of the Stanford bunny mesh: frontal (a) and smooth frame field (b) approaches.

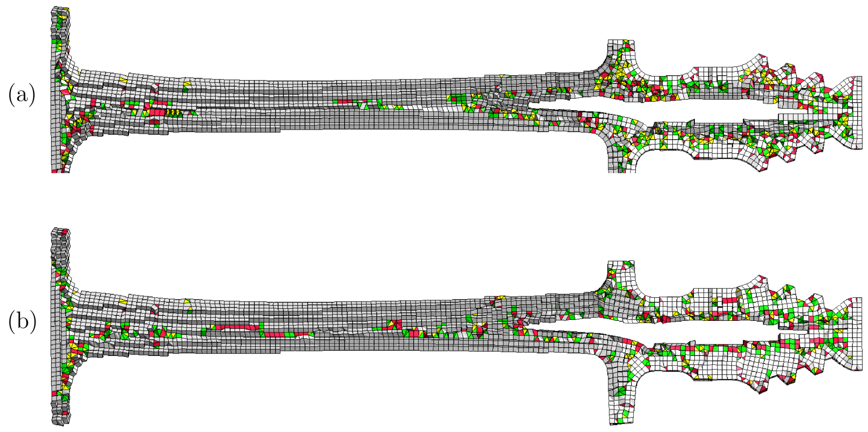

Figure 11: Cut view of the hollow turbine blade mesh: frontal (a) and smooth frame field (b) approaches.

boundaries of the bunny are more chaotic.
Indeed, for the dome, using the new (smoothness-based) point insertion only leads to 1% improvement in volume ratio and the frame field smoothing alone leads to 2.5% improvement. On the contrary, for the bunny, the new insertion leads to 8% improvement and the frame smoothing only to 6%. This is mostly due to the fact that the frontal algorithm does a better job on geometries with straight faces, while the non-smoothed initial frame field is worse with geometries presenting concavities as the dome. When combining the smoothing with the new insertion algorithm, we obtain a huge improvement of 7 to 18% in volume ratio.

Concerning the computational times, all computations show that the smoothness-based insertion is a bit slower. This was predictable since inserting elements in a ordered set has a larger cost than simply inserting at the end of a queue. But we also observe that, if the smoothing and insertion take more time, the tetrahedra recombination time usually strongly decreases. This is due to the fact

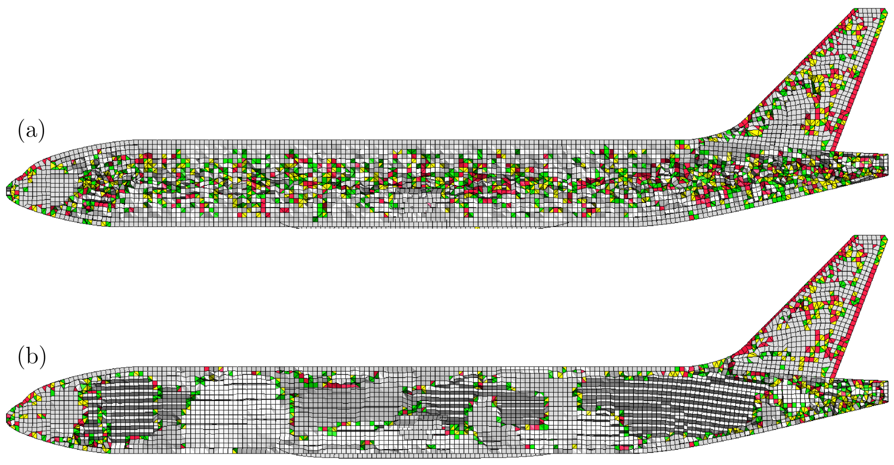

Figure 12: Cut view of the aircraft fuselage mesh: frontal (a) and smooth frame field (b) approaches.

that a better alignment of the points leads to a lower total number of elements, decreasing the amount of possible recombinations.

Geometry 8(c) is a hollow turbine blade, which involves a lot of thin parts (tends to benefit to the frontal algorithm) and presents concavities (decreasing the quality of the initial non-smoothed frame field). The consequence is that on the one hand, using the new insertion with a poor frame field does not yield much improvement, and on the other hand, the frontal algorithm does not benefit much of a smoothed frame field either. But again, combining the two leads to a volume ratio 5% larger. Cut views are depicted on Figure 11.
Geometry 8(d) is a mechanical piece involving straight faces and different parts not as thin as the blade. Therefore, this is a typical geometry where the simple frontal algorithm does a good job, the final improvement does not exceed about 2% for a computational time approximately 20% higher.

Geometry 8(e) consists in an aircraft fuselage with its tail. If boundaries are quite straight, it presents a large central volume to mesh: the new approach

Table 3: The times t_b, t_i and t_r are the times (on a single laptop computer) for background mesh and smoothing operations, for point insertion and for tetrahedra recombination respectively, t_t is the total time for meshing. For each test case, the frame field can be computed using the nearest neighbor (NN) or the smoothing iterative procedure (Sm), and the insertion algorithm can be either frontal using a queue (Fr) or frame field-based using an ordered set (FB).

Figure	Nb. elements	Time (s)				Hex ratio (%)	
		t_b	t_i	t_r	t_t	Nb	Vol
Fuselage (Fig.12)							
(AF)	95177	9.3	13.1	80.9	105.5	44.2	78.8
(AS)	94002	8.9	16.1	83.1	117.5	44.9	79.3
(SF)	91810	84.5	12.9	88.1	184.6	49	81.4
(SS)	77025	94	15.7	60.2	175.3	57.2	86.3
Naca (Fig.8(f))							
(AF)	24247	1.5	5.1	17.5	27.6	76.4	93.2
(AS)	25000	2	6	17.7	29.6	73	92.2
(SF)	22913	2	3.5	13.1	21.2	80.7	93.9
(SS)	24898	2	4.4	15.9	25.1	74.1	92.3

again leads to an improvement of 8% for a time increased by 70% (Table 3). On cut views of Figure 12, we see that, obviously, the frontal algorithm gives a nicer hexahedral mesh close to the fuselage straight boundary, but has issues when getting closer to the center.

Finally, geometry 8(f) is a simply extruded naca profile: the two approaches gives approximately the same result (less than 1% difference, which becomes non significant). We observed that, by imposing a smaller required mesh size, both methods reached a volume ratio of 97%, which confirms that the simple frontal algorithm gives good results for such simple geometries.

Note finally that the memory footprint did never exceed 800 Mb on the larger geometries.

5. Conclusions

A first iterative method has been proposed to obtain a smoother three-dimensional frame field. A second method was proposed to insert new points preferentially away from the geometric singularities.
This new frame field-based approach has revealed either as efficient or much more efficient than the previous frontal indirect approach, depending on the geometries, for a reasonable additional computational cost ranging from 0 to 70%. If each isolated method only leads to minor improvements, we observed that combining the two methods leads to large benefits in terms of hexahedra ratio. The larger improvements, up to 20%, have been observed on geometries presenting a large volume compared to their surface.

[1] R. Biswas, R. C. Strawn, Tetrahedral and hexahedral mesh adaptation for cfd problems, Applied Numerical Mathematics 26 (12) (1998) 135 – 151.

[2] M. A. Puso, J. Solberg, A stabilized nodally integrated tetrahedral, International Journal for Numerical Methods in Engineering 67 (6) (2006) 841–867.

[3] T. Carrier Baudouin, J.-F. Remacle, E. Marchandise, F. Henrotte, C. Geuzaine, A frontal approach to hex-dominant mesh generation, Advanced Modeling and Simulation in Engineering Sciences 1 (1) (2014) 1–30. doi:10.1186/2213-7467-1-8.

[4] S. Yamakawa, K. Shimada, Fully-automated hex-dominant mesh generation with directionality control via packing rectangular solid cells, Int J Numer Meth Eng 57 (2003) 2099–2129.

[5] E. Tomita, A. Tanaka, H. Takahashi, The worst-case time complexity for generating all maximal cliques and computational experiments, Theoretical Computer Science 363 (2006) 28 – 42.

[6] J.-F. Remacle, J. Lambrechts, B. Seny, E. Marchandise, A. Johnen, C. Geuzaine, Blossom-quad: a non-uniform quadrilateral mesh generator using a minimum cost perfect matching algorithm, Int J Numer Meth Eng accepted.

[7] J.-F. Remacle, F. Henrotte, T. Carrier-Baudouin, E. Bechet, E. Marchandise, C. Geuzaine, T. Mouton, A frontal delaunay quad mesh generator using the l norm, International Journal for Numerical Methods in Engineering 94 (5) (2013) 494–512.

[8] B. Levy, Y. Liu, l_p centroidal voronoi tessellation and its applications, in: H. Hoppe (Ed.), ACM Transactions on Graphics, Los Angeles, 2010.

[9] T. Carrier Baudouin, J.-F. Remacle, E. Marchandise, J. Lambrechts, F. Henrotte, Lloyd's energy minimization in the lp norm for quadrilateral surface mesh generation, Engineering with Computers 30 (1) (2012) 97–110.