INTERNATIONAL JOURNAL FOR NUMERICAL METHODS IN BIOMEDICAL ENGINEERING
Int. J. Numer. Meth. Biomed. Engng. (2015); e02725
Published online in Wiley Online Library (wileyonlinelibrary.com). DOI: 10.1002/cnm.2725

Mixed-element Octree: a meshing technique toward fast and real-time simulations in biomedical applications

Claudio Lobos*,† and Eugenio González

Departamento de Informática, Universidad Técnica Federico Santa María, Santiago, Chile

SUMMARY

This article introduces a meshing technique focused on fast and real-time simulation in a biomedical context. We describe in details our algorithm, which starts from a basic Octree regarding the constraints imposed by the simulation, and then, mixed-element patterns are applied over transitions between coarse and fine regions. The use of surface patterns, also composed by mixed elements, allows us to better represent curved domains decreasing the odds of creating invalid elements by adding as few nodes as possible. In contrast with other meshing techniques, we let the user define regions of greater refinement, and as a consequence of that refinement, we add as few nodes as possible to produce a mesh that is topologically correct. Therefore, our meshing technique gives more control on the number of nodes of the final mesh. We show several examples where the quality of the final mesh is acceptable, even without using quality filters. We believe that this new meshing technique is in the correct direction toward real-time simulation in the biomedical field.

Received 23 August 2014; Revised 15 April 2015; Accepted 17 May 2015

KEY WORDS: mixed-element mesh; adaptive meshing; refinement patterns; surface representation; real-time simulation; finite element method; element normalized scaled Jacobian

1. INTRODUCTION

In the last few years, several advances have been made in biomechanics in order to increase the accuracy and accelerate computational time of a simulation. The employed meshes and the techniques that produce them are a key term to accomplish fast and accurate results. We can identify two main streams with respect to the goal of the simulation:

- Accurate results, where time is not the dominant variable. In this case, the goal is to predict the outcome with high precision, and it is regularly used when we do not know what the outcome should be like. For these cases, the mesh tends to be extremely dense in order to fully understand the process being simulated.
- Fast results, where time is the dominant variable and precision is secondary. Here, we employ elements of different sizes, and our concern is to let the finest elements, with the highest quality, in the regions where the focus of the simulation is.

The work to be introduced here is in the second context: enable fast computation of biomedical applications, in order to be able to perform real-time simulations like surgery or obtain quick results like in the case of diagnosis of common procedures.

One tool is key to achieve this goal: refinement and coarsening. This tool enables mesh adaptation, which is capable of focusing more nodes in the interest zone of the simulation while leaving less nodes in other regions. Mesh adaptation is, thus, responsible for accelerating overall computational

*Correspondence to: Claudio Lobos, Departamento de Informática, Universidad Técnica Federico Santa María, Av. Vicuña Mackenna 3939, 8940897, San Joaquín, Santiago, Chile.
†E-mail: clobos@inf.utfsm.cl

times in a simulation. Once we have reached the target quantity of nodes, the natural question is if we can achieve accurate representation of the surface without increasing the quantity of nodes. Here, the use of surface patterns [1] that tends to better represent curved domains (like in the case of the biomedical field) is a key feature that may help us to increase the quality of the mesh while representing the boundary, the best we can with the given constraints.

The Octree technique [2, 3] is one of the most used techniques to concentrate nodes in a given region. Several previous techniques have proposed to perform the transitions between coarse and fine regions using only hexahedra [4–9]. On the other hand, Conti *et al*. [10] introduced some mixed-element patterns for transitions between coarse and fine regions for simulation with the finite volume method. More recently, the work of Nicolas and Fouquet in [11] has extended some transition patterns, composed of mixed elements, to the finite element (FE) method. However, their work is meant to be used when a hexahedral mesh already exists, and a greater refinement must be performed over it.

One advantage of using mixed elements is the reduction in the number of nodes to be inserted to manage the transition, and also that each time an octant is split, it is split in eight new octants, instead of the 27 octants needed when using only hexahedra. Therefore, we believe that mixed elements allow a greater control on the number of nodes of a mesh, unlike a mesh created only by hexahedra. This statement will be analyzed in Section 5.

Moreover, none of the mentioned works have implemented all possible cases to manage transitions between coarse and fine regions. They all implement what they consider the most common cases, and if a given configuration is not in the set of patterns, they split the octant. This is performed until all transition octants have a pattern to manage their particular configuration, which increases unnecessarily the amount of nodes throughout the mesh.

The goal of this work is to design a new meshing algorithm that performs more accurate control over the quantity of nodes, managing transitions with mixed elements, and adding as few nodes as possible, while achieving congruent topology throughout the mesh. We believe a better solution, than existing algorithms, can be achieved by implementing all transition patterns for an Octree technique. Finally, the use of new patterns that better represent curved domains can be a powerful tool to combine with; thus, achieving a mesh with as few nodes as needed, and with acceptable representation of the surface governed by the current amount of nodes. We believe that this type of mesh will be better prepared for a real-time biomechanical simulation.

2. MESHING TECHNIQUES BASED ON THE OCTREE STRUCTURE

The Octree was conceived by Yerry and Shephard in [2] as an extension of the Quadtree, introduced by Finkel and Bentley in [12]. The Octree is a recursive data structure that is composed of octants. An octant is a region of the space that has null volume intersection with any other octant, except for its sons. In most of the cases, the octant is simply a cube; however, as a concept, it could be of any geometry. The only constraint is that an octant must have a splitting operation, which will result in a constant number of new octants. Any meshing technique based on the Octree must count with a function that will decide if an octant must be split or not. In most cases, this function is governed by a level of error in the representation of the input geometry's boundary. As a result, the output volume mesh will be a non-conformal mesh, meaning that it is irregular in terms of element sizes.

In order to build a conformal mesh from the previous state, it is necessary to manage the transition between coarse and fine regions. An octant will be considered as 1-irregular when all of its neighbors were split at most one more time than it. The Octree technique states that all the octants that are not 1-irregular shall be split until this condition is fulfilled throughout the mesh. An example of this is shown in Figure 1.

There is a whole family of meshing techniques that split the octant in 27 new octants each time more refinement is needed [4–9]. By doing this, a set of the most common configurations can be employed to produce a transition between coarse and fine regions, using only hexahedra.

Another type of solution is to split octants in eight new octants whenever refinement is needed. In order to perform the transition, it is now necessary to introduce other types of elements, like tetrahedra, pyramids, and wedges (prisms). To our knowledge, they were first introduced in the

DOI: 10.1002/cnm

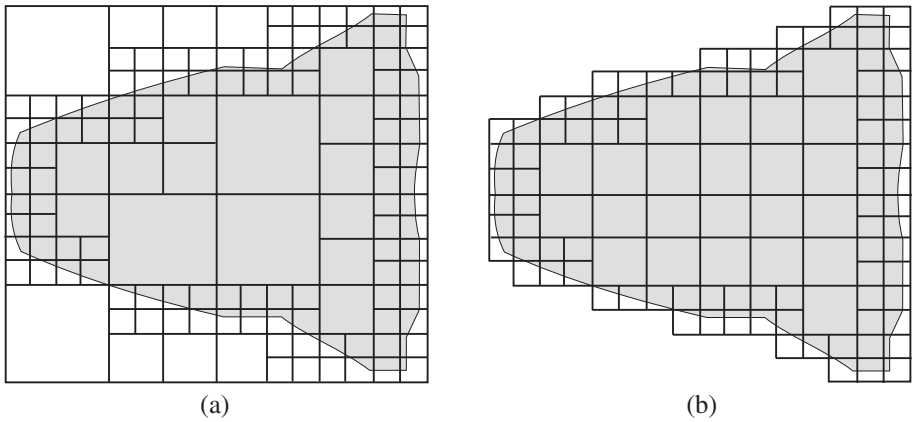


Figure 1. (a) Octree mesh and (b) 1-irregular mesh.

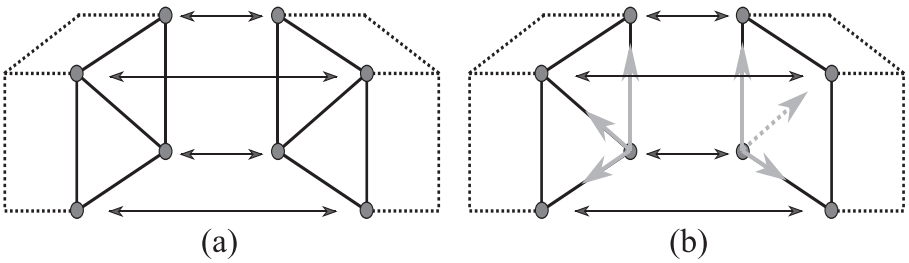


Figure 2. (a) Well-structured elements and (b) invalid configuration.

work of Conti *et al*. in [10], where a subset of all possible combinations were employed to enable fast simulation of semi-conductor devices with the finite volume method. The patterns in this work were used only in a medium step, because the numerical integration was finally carried out in the Voronoi diagram (the dual of a Delaunay tessellation [13]). This small detail allowed them to use patterns where the topology was not correct, because the Voronoi diagram of a square face and the diagram of same square face represented by two triangles is exactly the same. For this reason, if two wedges (prisms) representing a cube and a hexahedron are neighbors, they do not present any inconvenience for the Voronoi diagram. This can be seen in Figure 2. Note that this would only work if the simulation is run finally in the dual mesh, not in the one that you are currently building with the Octree.

Recently, the work of Nicolas and Fouquet in [11] introduced a subset of equivalent transition patterns to be used directly in a numerical simulation (not in the dual mesh). The main idea of this work was to, starting from an existing hexahedral mesh, split each hexahedron in eight new cubes, where more precision was needed by the simulation. Then, manage the transition with the existing patterns. If a pattern does not exist for a given configuration, continue splitting neighbor elements until a solution can be found.

After analyzing the aforementioned algorithms, we can say that, to our knowledge, no previous work has implemented the whole set of possible transition patterns, either for 8-element or 27-element split strategies. Therefore, our goals will be the following:

- implement all the patterns that allow to manage 1-irregular configurations in an Octree-based technique;
- analyze the quality of our solution where the goal will be to add as few nodes as possible;
- create an algorithm that will be useful to allocate a greater density of nodes and elements where the focus of the simulation is. With the idea that less nodes elsewhere will accelerate computational time, enabling real-time simulation; and

DOI: 10.1002/cnm

- be able to represent complex surfaces like in the case of biomedical applications and conserve acceptable quality of the elements.

Before entering in any further detail, we need to explain how the quality of our mixed elements can be measured. There are well-known metrics for tetrahedra and hexahedra, but there are not widely accepted ways to measure the quality of other element types. In Section 3, we propose a novel metric for quality measurement meant to be used by mixed elements. Section 4 explains in details the transition and surface patterns that will allow to comply with the aforementioned goals. Then, Section 5 explains why we believe our method has more control over node quantity than 27-element split strategies. In Section 6, the main proposed algorithm is detailed, and in this process, we use a liver as an example to illustrate each algorithm step. Then, in Section 7, we show the results for three different domains: brain, soft palate, and the bones of the foot. Finally, in Section 8, we discuss our results, showing that this algorithm will be an important alternative to generate a mesh in the context of biomedical application.

3. TOWARD A UNIFIED MEASUREMENT OF QUALITY FOR MIXED ELEMENTS

The measurement of quality has two purposes:

- to compare the mesh, either with other meshes produced for the same domain or with the own notion of a 'theoretically perfect' mesh; and
- to improve it, that is, acquire a 'better' mesh starting from a poor quality state by the use of mesh repairing algorithms.

In both cases, we need to count with the notion of a 'good' element. This task has been performed for the most common elements in meshes: the tetrahedron and the hexahedron. However, for other type of elements, this still remains as an open problem. In this section, we will introduce a new metric based on the scaled Jacobian (J_S), that will serve as a measurement of the quality of mixed elements shown in Figure 3.

In the case of the hexahedron, one quality metric based on the Jacobian is the scaled Jacobian (J_S), and it has been employed by many authors [8, 14–19]. Here, we say that the J_S of node i, that is, J_S^i, is the Jacobian of the tetrahedron formed by i and the three connected nodes to it in the element; this 'Jacobian' must be computed using the normalized vectors from i.

For example, consider the hexahedron of Figure 3(a) with nodes $0, \ldots, 7$ that will be labeled $n_0, \ldots, n_7$, respectively. If we are computing the scaled Jacobian for node a, that is, J_S^a, we define $d_i = n_i - n_a$ and $\hat{d}_i = d_i / ||d_i||$. Now, we can establish that

$$\begin{aligned} J_S^0 &= \hat{d}_4 \cdot \left(\hat{d}_1 \times \hat{d}_3\right) = \hat{d}_1 \cdot \left(\hat{d}_3 \times \hat{d}_4\right) = \hat{d}_3 \cdot \left(\hat{d}_4 \times \hat{d}_1\right) = 1 \\ &\neq \hat{d}_4 \cdot \left(\hat{d}_3 \times \hat{d}_1\right) = -1 \end{aligned} \tag{1}$$

With this formulation, it is clear that $J_S \in [-1, 1]$, and only positive values are acceptable for an FE simulation. Note that when computing J_R^n, the order of the normalized vectors in the formula

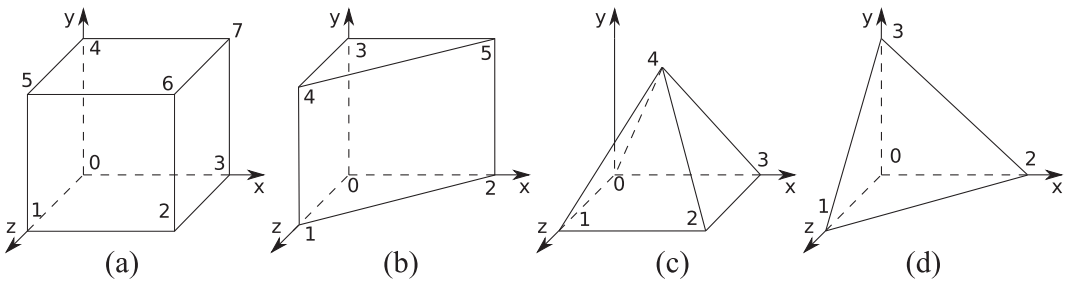


Figure 3. Basic elements: (a) hexahedron, (b) prism (wedge), (c) pyramid, and (d) tetrahedron.

DOI: 10.1002/cnm

$\hat{d}_i \cdot (\hat{d}_j \times \hat{d}_k)$ must follow the right-hand rule in the reference system. Otherwise, you will obtain negative values when the element is actually valid.

It is very important to keep in mind that J_S is a measurement of quality based on the Jacobian; it is not the Jacobian itself. The importance of these metrics is to establish when an element is invalid (< 0), when it is reasonably 'good' ($0.2 <$), and when it is perfect ($= 1$).

The J_S is not an accurate way of computing the quality of a tetrahedron. If we compute J_S for an equilateral tetrahedron, we will obtain $J_S^i = \sqrt{2}/2 \approx 0.7, \forall i = \{0, 1, 2, 3\}$; however, we would expect the value $+1$ for each node. One way to obtain similar values is to normalize the J_S regarding element type. Here, we propose the element normalized scaled Jacobian (J_{ENS}) as follows:

$$J_{ENS}^n = \begin{cases} (1 + k^e) - J_S & \text{if } J_S > k^e \\ J_S / k^e & \text{if } -k^e \leqslant J_S \leqslant k^e \\ -(1 + k^e) - J_S & \text{if } J_S < -k^e \end{cases}$$

$$E_q = \begin{cases} \min\left\{J_{ENS}^i\right\} & \text{if } \forall J_{ENS}^i > 0 \\ \max\left\{J_{ENS}^a\right\} : J_{ENS}^a < 0 & \text{if } \exists J_{ENS}^i < 0 \end{cases}$$

where k^e is a constant value $k^e = k^T = k^P = \sqrt{2}/2$ for the tetrahedron and the pyramid, $k^e = k^R = \sqrt{3}/2$ for the prism, and $k^e = k^H = 1$ for the hexahedron. The constants values (k^e) for each element type are computed by building an equilateral element (all edges of equal length), and the J_S is measured at the nodes. Then, we define the quality of an element (E_q) as the minimum value of J_{ENS} at one of its nodes, if all J_{ENS} at the element are positive, and as the maximum value of J_{ENS} among its negative nodes, when at least one is negative.

In the case of the tetrahedron, we can compare the J_{ENS} with other quality metrics. One particular measure employed by several softwares and libraries like CUBIT, CSIMSOFT, SCIRUN, VERDICT, among others, is the so called aspect ratio Gamma (ARG), described in [20]. The ARG, when computed with signed volume, detects the most common cases of invalid and poor quality tetrahedra: flat, needle, and inversion (inside out element). Examples of these types of tetrahedra are shown in Figure 4. The ARG is defined as follows:

$$R = \left(\frac{1}{6}\sum_{i=0}^{5} ||l_i||^2\right)^{\frac{1}{2}} \qquad \Rightarrow \qquad \text{ARG} = \frac{R^3\sqrt{2}}{12 \cdot V} = \frac{R^3}{8.48528 \cdot V}$$

where $l_i, \forall i = \{0, \ldots, 5\}$ are its edge lengths and V is the volume of the tetrahedron. Now, we have to put this metric in our reference scale because ARG $\in [1, \infty)$. First, because we consider signed volume, this section of the equation will detect inverted elements. Now, to be in the same scale as the J_S, we must use ARG^{-1}. We see that ARG$^{-1} \in [-1, 0] \cup [0, 1] = [-1, 1]$, which is coherent with the scale defined for the J_S.

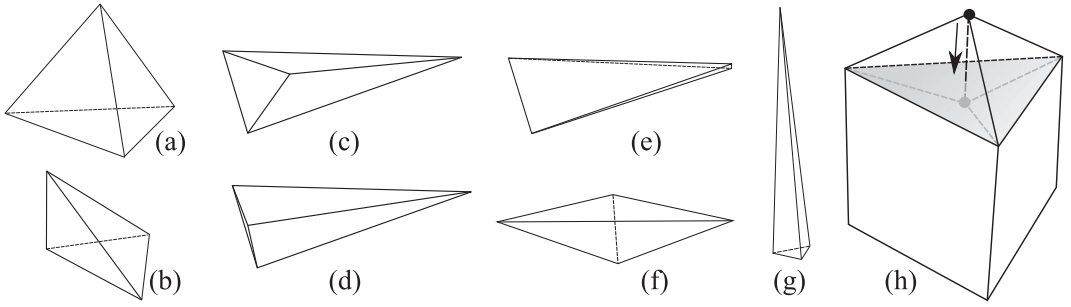


Figure 4. Different tetrahedra: (a) equilateral, (b) rectangle, (c) flat 1, (d) flat 2, (e) wedge, (f) sliver, (g) needle, and (h) how an inverted tetrahedron can be formed. Due to node displacement, the apex may pass through the triangular base, which will lead to a negative volume.

DOI: 10.1002/cnm

Table I. Comparison of different quality metrics for valid tetrahedra.

		Equilateral	Rectangle	Flat 1	Flat 2	Wedge	Sliver	Needle
ARG^{-1}		1	0.769	0.004	0.004	0.003	0.005	0.021
J_S	min	0.707	0.5	0.003	0.002	0.002	0.002	0.005
	max	0.707	1	0.008	0.008	0.866	0.002	0.57
J_{ENS}	min	1	0.707	0.003	0.003	0.002	0.003	0.007
	max	1	0.707	0.012	0.011	0.841	0.003	0.806

In Table I, values from the three quality metrics are contrasted for the set of tetrahedra shown in Figure 4: the ARG^{-1} is computed for each element, and then the minimum and maximum value for J_S and J_{ENS} are also shown for each element. It is clear that the minimum value for these metrics are in direct correlation with ARG^{-1}.

With respect to other types of elements, different from the tetrahedron and the hexahedron, there is little to say in terms of quality measurement. Even though there are some geometry-based quality metrics, it would be necessary to have a metric that took into account the fact that these elements must coexist among tetrahedra and hexahedra. For this reason, here, we propose an extension of J_{ENS} for the pyramid and the prism (wedge).

Note that among all elements of Figure 3, all the nodes are connected to other three nodes within the element, except for the top node of the pyramid (node 4 in Figure 3(c)). To measure the quality of the top node t, we compute J_S^t for the four possible tetrahedra formed by t and its neighbors in the base face. We assign to J_S^t the minimum among positive values. If there are no positive values, then the maximum among the negative values is assigned to J_s^t. This is because sometimes, one of the four possible values is negative, and this means that one of the base nodes is invalid; however, the top node is well located.

Finally, one thing about the prism is that if we drastically increase the distance between its two triangular faces, and they are equilaterals, this element will be perfect in terms of J_{ENS}. This will also happen for the hexahedron: any hexahedron with 90° angles will be considered as the perfect element by the J_S and, therefore, by the J_{ENS}. Fortunately, the meshing technique described in this article does not tend to create elements with this shape. The worst ratio for an hexahedron would be equivalent to put two regular hexahedra next to each other. In the case of prisms, this ratio is even better. More discussion over J_{ENS} can be found in [21].

4. MIXED-ELEMENT PATTERNS

4.1. Transition pattern

The goal of a transition pattern is to ensure correct mesh topology between regions of a different refinement level (RL). To do so, the mesh must be 1-irregular. As explained in Section 2, this is equivalent to the following statement: every edge of an octant (in the current state of the Octree) is split at most in two sub-edges. If all the edges of a square face are split, the center of the face may also have an inserted node.

The patterns introduced in this work are meant to insert no new nodes, except when the task of subdividing the space using the basic elements is impossible. By basic elements, we refer to the hexahedron, prism (wedge), pyramid, and the tetrahedron, which were shown in Figure 3.

To illustrate the idea of a transition pattern, let us consider the situation presented in Figure 5(a), where B is the only neighbor of octants A and C. It can be seen that octant A has one more RL than the other two, and all the octants are 1-irregular; in particular, octant B is the only one that needs a transition pattern, as it counts with four splitted edges and one central node inserted in a face; therefore, it counts with five extra nodes inserted in one face.

In this case, the solution is to replace the current hexahedron of the octant with five pyramids and four tetrahedra: one big pyramid that will be the new neighbor of octant C, four little pyramids that will be the new neighbors of the cubes of octant A, and four tetrahedra that will fill the void space

DOI: 10.1002/cnm

between the pyramids. The result can be seen in Figure 5(b), where the white faces belong to the visible pyramids, while the dark faces belong to visible tetrahedra.

Once octant B is split, the mesh consisting of octants A, B, and C, is considered as topologically correct. This mesh consist of nine hexahedra: one of them is in octant C and the rest belong to A, and five pyramids and four tetrahedra associated to octant B. To do so, it was not necessary to insert new nodes; however, nine elements were used to replace the hexahedron of octant B, which leads to a total of 18 elements instead of the 10 that were not topologically correct. This is important because in Section 7, it will be shown that the use of these transition patterns increases, naturally, the overall number of elements.

Figure 6 shows all possible configurations of nodes inserted in a face and also how to split the face in each case. It is easy to see that in the example shown in Figure 5, the following patterns were employed over octant B: the pattern of five inserted nodes shown in Figure 6(g), to connect with neighbor octant A, and four instances of the pattern shown in Figure 6(b), for each face that was not shared with any other octant.

The next step was to build all patterns. We first programmed a code capable of producing all combinations of face patterns shown in Figure 6. Then, we removed incongruent combinations, meaning that if an edge was split, all incident faces to that edge managed it as a divided edge. Finally, we considered rotation in all axes to detect equivalences between different combinations.

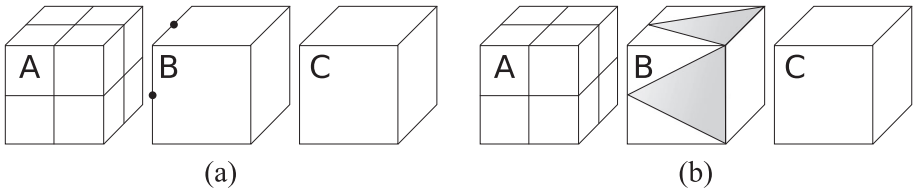


Figure 5. (a) 1-irregular mesh where octant B is the only one that needs a transition pattern and (b) the resulting mesh.

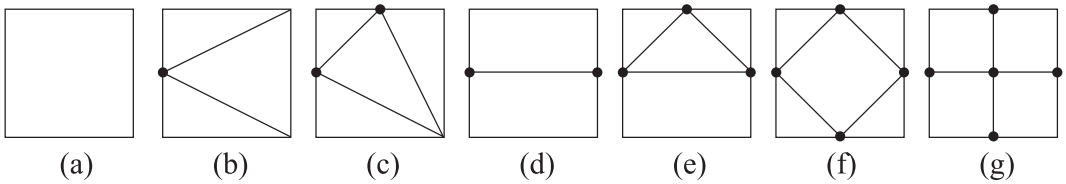


Figure 6. All possible cases for nodes inserted in edges or face central position.

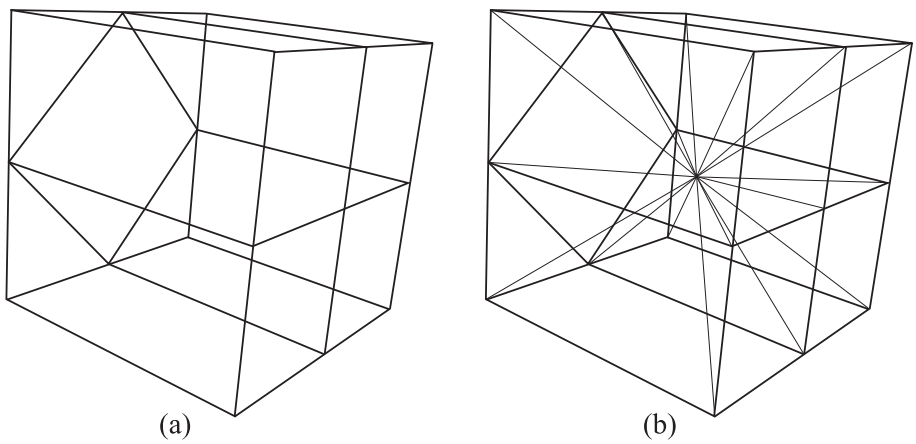


Figure 7. Pattern that needs the insertion of cube's central node: (a) face splitting and (b) internal elements.

DOI: 10.1002/cnm

Table II. The J_{ENS} quality values over the whole set of transition patterns.

	Tetrahedron	Pyramid	Prism	Hexahedron	Overall
Worst	0.172	0.23	0.516	1	0.172
Best	1	1	0.923	1	1
Average	0.498	0.477	0.721	1	0.509
Quantity	2378	1511	277	35	4201

The final amount of patterns is 325, and they are specified in the technical report [22]. Note that this amount could have decreased if we had considered mirroring operations.

Then, we proceeded to implement all of the patterns. However, for some configurations, it was impossible to split the octant in the basic elements shown in Figure 3. In these cases, the only option was to add the central node of the cube. With this extra node, we could now add several internal diagonals and accomplish the task of discretizing the element, regarding the face patterns of Figure 6. An example of this can be seen in Figure 7, where the original cube is split in 13 pyramids and four tetrahedra.

In the 260 cases out of the 325 patterns, it was possible to produce a subdivision of the cubes without inserting the middle node. As mentioned before, the focus of the transition patterns is to reduce the overall quantity of nodes; therefore, it is expected that they themselves do not involve the insertion of a large quantity of nodes. Indeed, 65 patterns need the insertion of only one extra node in order to accomplish with the constraints imposed by the face patterns of Figure 6. With these patterns, it is now possible to manage all the cases of 1-irregular elements.

Table II shows the J_{ENS} quality statistics for all element types in the whole set of patterns. The goal of this table is to illustrate that these transition patterns do not count with perfect quality, even in a stationary state before suffering any deformation. Indeed, these patterns were designed with the idea of increasing as less as possible the number of nodes, not with the idea of achieving good quality. It is important to retain these values as the lower quality threshold when we focus on inserting as few nodes as possible in the output volume mesh.

Note that the best quality for a prism in Table II is not 1. This is because the J_{ENS} reserve the value 1 for a prism with equilateral triangular faces. Therefore, this means that our patterns do not count with the perfect equilateral prism.

4.2. Surface patterns for all type of elements

A previous work, presented in [1], allows to replace all hexahedral elements intersecting the surface of the input domain by mixed elements. To perform this task, we need the set of triangles that conform the surface of the input domain Ω^s. Using the signed distance algorithm introduced in [23], we determine if a node is inside or outside Ω^s. The same algorithm is employed to find the projection of a node onto Ω^s.

This approach has shown to be better prepared when representing curved domains as it reduces quality issues of some elements. The drawback of this replacement is that sometimes, this will end up in a lack of boundary representation. Figure 8 shows an example of this: a hexahedron that, otherwise, would present quality issues is replaced with a prism (wedge). By doing this, we avoid quality issues. The drawback is that with this, we decrease boundary representation. Note that for a general case, we cannot ensure that all quality issues will be avoided by the use of these surface patterns.

In the present work, we will extend this approach to all types of elements, as not only hexahedra may be found at the border of the output mesh but also pyramids and prisms. In order to explain this, Figure 9 shows the convention to be employed over each quadrilateral face. For instance, Figure 9(b) shows that a square face with only one node inside the domain results in two triangles regarding the segmented line. When acting over an element, like in the case of Figure 8(a), it will end up splitting the hexahedron into two prisms, and one of them is completely outside of the domain, so the final result is only one prism as shown in Figure 8(c).

DOI: 10.1002/cnm

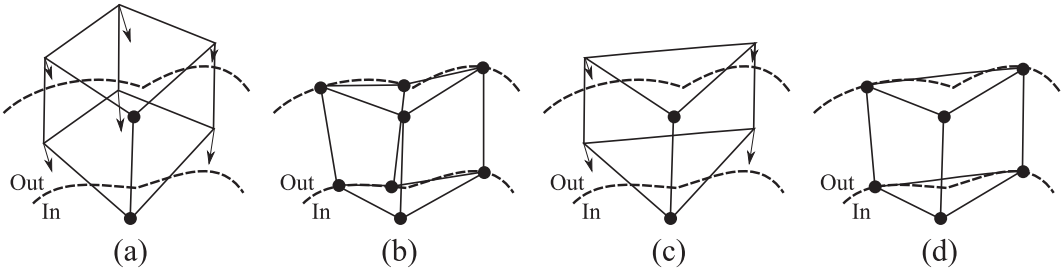


Figure 8. Using surface patterns: (a) hexahedron intersecting the surface, (b) the projection of outside nodes into the boundary produces a poor quality element, (c) the hexahedron is replaced with an appropriate element for this configuration, and (d) the projection of outside nodes now results in a high-quality element.

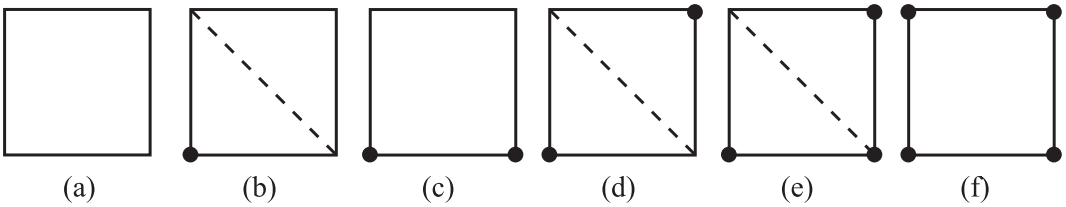


Figure 9. Convention for splitting a square face into triangles. Inside nodes are represented by circles.

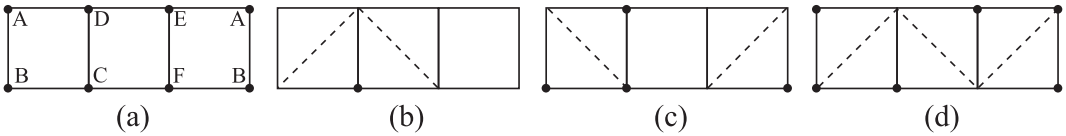


Figure 10. Surface patterns for a prism: (a) the three quadrilateral faces of a prism lined up with its six nodes, (b), (c), and (d) are examples of diagonal insertion.

The case of a pyramid is simple, as it counts with only one quadrilateral face. Therefore, the subdivision of the pyramid into two tetrahedra will depend on the number of inside nodes in the base face, following the rules presented in Figure 9. Note that for some cases, as shown in the example of Figure 8, the pyramid may be replaced with only one tetrahedron.

In the case of the prism, there are more options to analyze. This element counts with three quadrilateral faces, as shown in Figure 10(a), and regarding the inside/outside state of its nodes, the outcome may result in one of the following three cases:

- One prism. This happens when no diagonal was inserted. This will occur when no node is inside or all of them are, and in some other cases, like for instance when nodes B, C, F or A, B, C, D are inside.
- One pyramid and one tetrahedron. This happens when two diagonals are inserted. Note that those two diagonals will always share a node. This occurs when, for instance, node C is inside, as Figure 10(b) shows. Here, node D is shared by the two diagonals. Another example is when nodes B, C, D are inside, as shown in Figure 10(c). Here, node A is shared by diagonals.
- Three tetrahedra. This happens when three diagonals are inserted. Note that among the three of them, two nodes will always be shared. An example of this is when nodes A, B, C, E are inside, as Figure 10(d) shows. In this example, nodes D and F are shared by diagonals.

5. CONTROL OVER NODE QUANTITY WITH OUR STRATEGY

In the following lines, we will demonstrate how it is possible to gain more control over node quantity in the mesh using the 8-element split process rather than the 27-element strategy. Figure 11(a) shows

DOI: 10.1002/cnm

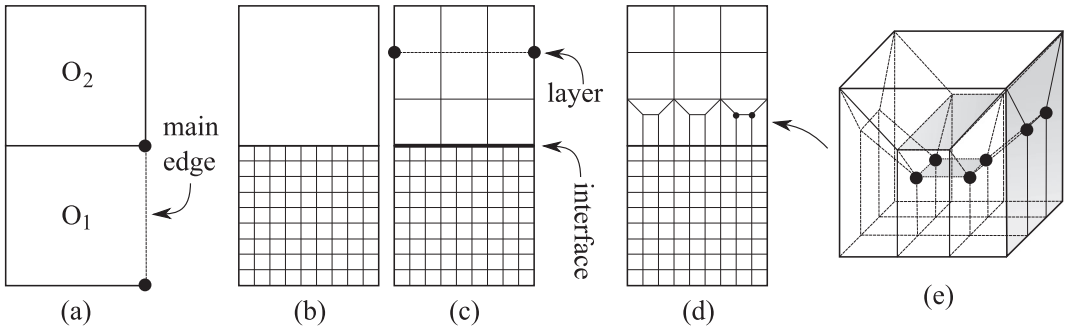


Figure 11. Example of using 27-node split strategy: (a) octant O_1 will be refined to level 2, (b) end of refinement process, (c) end of making mesh 1-irregular, and (d) insertion of transition pattern shown in (e).

the starting point for both types of refinements, where the idea will be to split the bottom octant (O_1), with different RLs, and at the top octant (O_2), add just the necessary nodes to achieve a congruent topology.

There are three main steps that may add nodes to the mesh in these examples. The direct refinement of an octant, making the mesh 1-irregular and the nodes added by the use of transition patterns. Let us first determine at which rate the 27-element splitting strategy adds nodes due to octant refinement.

5.1. Node insertion by direct octant refinement

Each time an octant must be split, one edge of it adds two nodes, and therefore, one edge becomes three new edges. The recurrence that establishes how many nodes we have at splitting iteration $n+1$ (i^{n+1}) will be the number of nodes we had in previous iteration (i^n), plus two nodes for each edge at i^n. Mathematically speaking, the number of points given at i^{n+1} is $P_{n+1} = P_n + 2 \times (P_n - 1)$, with border condition $P_0 = 2$. Using ordinary generating functions [24], it is easy to see that $P_n = 3^n + 1$. This means that at RL $= 1$, we have $3^1 + 1 = 4$ nodes in a 'main edge' (Figure 11(a)), and for RL $= 2$, we have $3^2 + 1 = 10$ nodes in a main edge. Now, with this quantity, we can establish that for 2D, the quantity of nodes in the 'face' of O_1 will the number of nodes at one main edge to the power of 2. This is 100 nodes for an RL $= 2$. Consequently, an RL $= 2$ of an octant in 3D is the number of nodes in a main edge to the power of 3, that is, 1000 nodes. So this is the first part of the equation: if an O_1 must be refined to level n, it will need $(3^n + 1)^3$ nodes with this strategy.

5.2. Node insertion by making the mesh 1-irregular

To compute how many nodes will be inserted by making the mesh 1-irregular, let us use the concept of a layer shown in Figure 11(c), which in 3D refers to a square face (and in 2D to an edge). This process ends when the difference in RL between neighbors is not greater than one. In the example of Figure 11, O_1 has an RL $= 2$; therefore, O_2 must have at least RL $= 1$, and we need to split it. This would add four 'layers', but all the nodes of the layer labeled as 'interface' (between octants) in Figure 11(c) have already been inserted by the refinement of O_1. Now, the number of nodes at each layer for the example will be $(3^{i-1} + 1)^2$, where $i = 2$, that is, the RL of O_1. The total number of nodes by 1-irregular state will be $3 \cdot (3^{i-1} + 1)^2$. However, this is not the general case. If we force O_1 to an RL $= 3$, then we would have to perform the same task and also add more nodes to achieve 1-irregular property between neighbors of RL $= 3$ and RL $= 2$. An example of this can be seen in Figure 12(c), where two layers of RL $= 1$ have survived and the third layer of RL $= 1$ (from previous iteration) has turned into three layers of RL $= 2$.

A generalization of this when O_1 has an RL $= n$ would be the following:

$$\left(\sum_{i=1}^{n-2}(3^i + 1)^2 \cdot 2\right) + 3 \cdot (3^{n-1} + 1)^2 = (3^{n-1} + 1)^2 + 2\sum_{i=1}^{n-1}(3^i + 1)^2$$

DOI: 10.1002/cnm

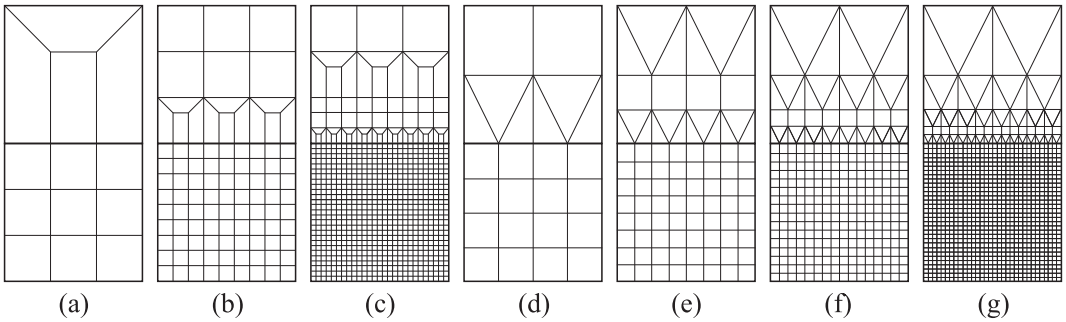


Figure 12. The refinement of the bottom octant using 27-node split in (a)–(c) and 8-node split (d)–(g). The refinement levels are (a) 1, (b) 2, (c) 3, (d) 2, (e) 3, (f) 4, and (g) 5.

This formula can be understood as two layers of RL $= 1$ will survive, plus two more of RL $= 2$, and so on until an RL $= n-2$; finally, we need to add the three layers of RL $= n-1$. Recalling that a layer of RL $= i$ adds $(3^i + 1)^2$ nodes. With this calculus, we have already counted all the nodes in the example of Figure 11(c), and we can start applying the transition pattern of Figure 11(e) into the current state of the mesh.

5.3. Node insertion by transition patterns

Now, transition patterns must be used to achieve correct topology throughout the mesh. We will need to use nine times the transition pattern of Figure 11(e) to achieve correct topology between regions of RL $= 2$ and RL $= 1$, and then, 27 between RL $= 3$ and RL $= 2$, and so on. As each instance of the transition pattern has four internal nodes, this will add $\sum_{i=1}^{n}(3^{i-1})^2 \cdot 4$ nodes. Now each instance counts with four faces, and each one adds two extra nodes as shown in Figure 11(e); however, several of these faces are shared by two octants. The number of these faces corresponds to the number of edges in the layer next to the interface. Let us recall that in 3D, this layer is a square face with an RL $= n-1$. Let us first count the number of horizontal edges in the layer. In one main edge, we will find 3^{n-1} edges. The number of these horizontal main edges corresponds to the number of nodes in a main edge over the same interface, that is, $3^{n-1} + 1$. Using the fact that the number of horizontal equals the number of vertical edges and each one of the transition faces adds two nodes, the number of nodes added only by the faces of this transition pattern is $\sum_{i=1}^{n}(3^{i-1} + 1) \cdot 3^{i-1} \cdot 4$. So the total amount of nodes, inserted only by the use of this transition pattern, is $\sum_{i=1}^{n}(3^{i-1} + 1) \cdot 3^{i-1} \cdot 4 + (3^{i-1}) \cdot 4$.

We can finally say that using the 27-element split strategy, the amount of nodes in the final mesh when O_1 is refined to level n is

$$[(3^n + 1)^3] + \left[(3^{n-1} + 1)^2 + 2\sum_{i=1}^{n-1}(3^i + 1)^2\right] + \left[4 \cdot \sum_{i=1}^{n} 2 \cdot (3^{i-1})^2 + 3^{i-1}\right]$$

Note that a separation by square brackets was employed to illustrate the influence of the three detailed sections: direct refinement, making the mesh be 1-irregular, and applying the transition patterns.

5.4. Comparison between 27/8-element splitting strategies

Now, we can use the exact same logic to obtain the following formula when using the 8-element split strategy:

$$[(\mathbf{2}^n + 1)^3] + \left[(\mathbf{2}^{n-1} + 1)^2 + \mathbf{1} \cdot \sum_{i=1}^{n-1}(\mathbf{2}^i + 1)^2\right] + [\mathbf{0}]$$

The differences are highlighted in the formula and correspond to the following:

DOI: 10.1002/cnm

- We changed all values of three to two, because now splitting an edge creates two instead of three new edges.
- We changed factor two for one at the summatory of the 1-irregular section. This is because now only one layer 'survives' from previous iteration (instead of two for 27-element strategy).
- The transition pattern adds no new nodes; therefore, we changed the previous expression to 0.

Table III. Number of nodes used by 27-element and 8-element split strategies for different values of refinement level (RL).

27-element					8-element			
RL	Direct refinement	1-irregular	Transition patterns	Total number	RL	Direct	1-irregular refinement	Total number
1	64	4	12	80	1	27	4	31
2	1000	48	96	1144	2	125	13	138
3	2.2E+04	332	780	23064	3	729	38	767
4	5.5E+05	2584	6720	560672	4	4913	119	5032
5	1.4E+07	2.2E+04	59532	1.5E+07	5	35937	408	36345
6	3.9E+08	1.9E+05	532896	3.9E+08	6	274625	1497	276122
7	1.0E+10	1.7E+06	4787340	1.0E+10	7	2.1E+06	5722	2.1E+06
8	2.8E+11	1.5E+07	4.3E+07	2.8E+11	8	1.7E+07	22363	1.7E+07

In Table III, we can see a comparison between both methods in terms of node quantity. This table separates the quantity of nodes inserted by each of the algorithm steps to achieve a conformal non-regular mesh: direct refinement (of the bottom octant O_1), 1-irregular state of the mesh, and finally, the use of transition patterns. It may be seen as a convenient example toward 8-element split technique due to the fact that the chosen configuration does not require the insertion of extra nodes by the transition pattern. However, it is important to recall that, out of the 325 transition patterns in our set, only 65 add one extra node. For the rest, there is no need to insert new nodes. Moreover, to manage the transition between fine and coarse regions, the shown example is the most basic and one of the most common configurations in the process to obtain a conformal mesh.

On the other hand, we could even say that this example is convenient toward 27-element split strategy, because the only transition pattern needed exists in the set of implemented patterns. Let us recall that several implementations of 27-element splitting strategy (if not all), do not count with a set of transition patterns capable of managing all possible configurations between two neighbor octants that fulfill the 1-irregular property. Should the pattern not be found in the set, both neighbors should be refined to the same level. In the worst scenario, this will propagate to a level in which the entire mesh will be refined to the same level: a grid mesh. To imagine this case in terms of the example with only two octants, this would be like taking the amount of nodes needed by O_1 and multiplying it by 2 to obtain the final amount of nodes in the mesh. For instance, instead of the 1144 nodes for RL $= 2$ in Table III, the mesh would have 1000×2 nodes.

All of the aforementioned discussions enable us to state that our technique, based on 8-element split strategy, allows to gain more control over the quantity of nodes used by the common 27-element split strategies:

- The rate at which the nodes are inserted by direct refinement is slower:

$$\text{8-element: } \lim_{n\to\infty} \frac{(2^n+1)^3}{(2^{n-1}+1)^3} = 8 \qquad \text{27-element: } \lim_{n\to\infty} \frac{(3^n+1)^3}{(3^{n-1}+1)^3} = 27$$

  This means that if we increase in one level the refinement, we will tend to multiply by eight the number of nodes of the octants being refined, while in the other strategy, we will tend to multiply by a factor of 27.
- Something similar happens with the rate of insertion by making the mesh 1-irregular.
- Finally, with respect to transition patterns, our method counts with two advantages: in 260 cases, out of 325, we do not need to insert any extra node, and for the rest, they need each,

DOI: 10.1002/cnm

only one extra node. In contrast, the patterns for 27-element split add several nodes internally and at the faces of the octant. The second advantage is that every time the 27-element split strategy does not count with a pattern for a given configuration, they refine the octant one more level. And they will continue performing unnecessary octant refinement until they find a configuration for which a pattern exists. In our case, as we have implemented all possible cases (325), we will not propagate unnecessary refinement.

6. ALGORITHM BASED ON MIXED-ELEMENT PATTERNS

For the sake of comprehension, first, we will introduce our algorithm in simple words. Then, we will show it in pseudo-code, with more details on each of the lines that correspond to different procedures of our main algorithm.

There are two inputs to the algorithm: a surface mesh formed by triangles (Ω^s) and a set of regions of interest (RoI), each with an independent RL, that indicate where the refinement is needed. Once the refinement is reached, we need to split all octants that are not 1-irregular. An edge complies with the 1-irregular property when it has at most one node inserted between its two extreme nodes. An octant, defined by a hexahedron at this time, is said to be 1-irregular when all of its edges are 1-irregular. When this property is ensured throughout the mesh, we can apply the transition patterns wherever they are needed.

Now, we count with some octants that intersect the input surface Ω^s, meaning that some nodes of it are outside and others inside Ω^s. If we directly project outside nodes onto the Ω^s, we may produce poor quality or invalid elements. For this reason, we try to push out the elements that are almost outside Ω^s. Then, we apply the surface patterns to remaining octants intersecting Ω^s. Finally, we project outside nodes onto the surface. Eventually, some elements will be added if they help to decrease the staircase effect that may be produced at some regions.

Now that we have the big picture, Algorithm 1 shows all the steps to execute the aforementioned ideas. In the following lines, we give more details on each of the steps. We will use a liver as an example to better illustrate the algorithm.

Algorithm 1 Mixed elements meshing main algorithm

Require: input surface domain: Ω^s, set of regions of refinement: RoI_set.

```
1: mesh ← base_octree
2: while refinement is not reached do
3:     list_octants ← detect_octants_for_refinement(RoI_set)
4:     mesh ← split_octants(Ω^s, mesh, list_octants)
5: end while
6: while mesh is not 1-irregular do
7:     list_octants ← detect_not_1_irregular()
8:     mesh ← split_octants(Ω^s, mesh, list_octants)
9: end while
10: mesh ← apply_transition_patterns(mesh)
11: mesh ← remove_close_to_boundary_elements(mesh)
12: mesh ← apply_surface_patterns(mesh)
13: mesh ← shrink_outside_nodes(Ω^s, mesh)
14: mesh ← decrease_staircase_effect(Ω^s, mesh)
15: write_eoutput(mesh)
```

The algorithm will employ Ω^s to determine if we are inside or outside the domain. It will also be used to project some nodes onto it at the end of the algorithm. The only constraint over Ω^s is that there cannot be self-intersections because this would lead to inconsistency in the inside/outside test. Figure 13 shows two meshes of the liver. The first one presents a self-intersection as it can be seen in Figure 13(b). There are many useful and open source tools to overcome this type of

DOI: 10.1002/cnm

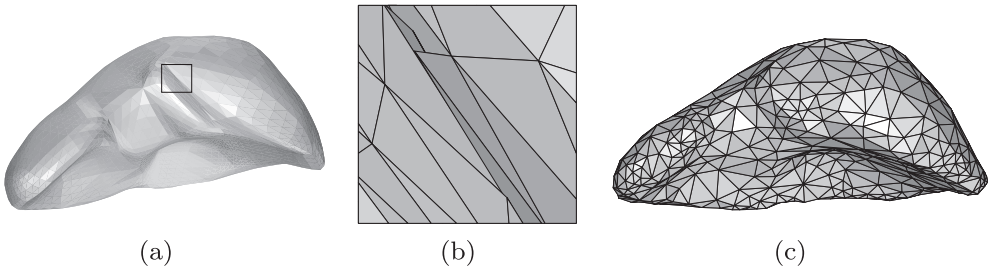


Figure 13. Two meshes of the liver: (a) mesh with self-intersection, (b) zoom to the self-intersection region, and (c) another mesh for the same domain, without self-intersection and with less triangles but of greater quality than the first mesh.

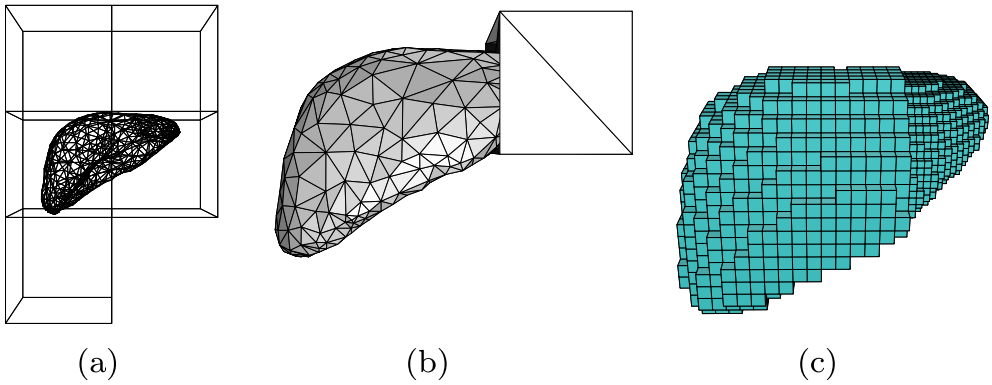


Figure 14. (a) Initial octants in the Octree structure for this example, (b) the triangulated surface Ω^s and a region of interest defined by a cube, and (c) the output of the Octree with the given constraints of refinement.

problems. In particular, we followed the idea introduced in [25], producing a mesh of superior quality and less triangles.

This meshing technique is based on the Octree; therefore, line 1 determines the roots of the Octree structure. Note that in this implementation, the root may be more than one octant. This idea allows it to be better adapted for non-cube-like structures; in other words, a cube may be a good starting point to mesh a prostate; however, it is not a good start for a femur bone; in which case, several aligned small cubes would be the best starting point. The particular starting point of the liver example is shown in Figure 14(a).

Before continuing with the explanation of the algorithm, it is necessary to establish the three different states of an element. Two of them are obvious: inside and outside of the domain (Ω). The third state is of an element that intersects the domain's boundary (Ω^s), meaning that it counts with at least one node inside and at least one node outside Ω. In the rest of the text, we will refer to this type of element as a surface element.

Then, lines 2–5 of Algorithm 1 allow to split each hexahedron in eight new ones until the required RL is reached. There are three options to set mesh RLs:

- Provide a set of RoI, where octants will be refined. Each RoI will be described by a surface mesh that intersects the input domain, and also, each RoI will be associated to a given level of refinement, which is the number of times a hexahedron is split in eight new ones. If an octant belongs to several regions, its refinement will be the maximum level among them.
- An overall surface RL, meaning that each octant that intersects Ω^s will be refined that number of times.
- An overall RL, which means that all octants in the mesh will be refined to the same level, or equivalently, all octants will be of the same size (a grid mesh).

DOI: 10.1002/cnm

To better explain our algorithm, we will use these three types of refinements at the same time over the liver example: overall RL = 3, overall surface RL = 4, and octants intersecting the region defined by the cube of Figure 14(b), must present RL = 5. The output of the Octree refinement can be seen in Figure 14(c).

It is important to highlight that the efficiency of the Octree technique is based on the local information of an octant. In other words, it is not only important to know if an octant intersects the surface but also what faces are intersected. In our implementation, to test intersection of octant O and triangle $t \in \Omega^s$, we first check if a node of t is inside O. This is efficient because the octant is currently a cube aligned with the axes; therefore, to test if a point is inside the cube, we only need to compare their coordinates. When the answer is negative, we test if a segment of t intersects O. In order to do so, we employ the 3D *Cohen–Sutherland* clipping algorithm [26, p. 113]. If the answer is still negative, we test if an edge of O intersects t. The last test is hardly used, due to the fact that the two previous tests allow us to detect most of the cases efficiently. Moreover, if an octant intersects no triangle, we test only one node of it against its parent's list of 'intersection faces'. If the node is outside, the octant is removed from the mesh. Otherwise, the octant and all its eventual sons are no longer tested for intersections, and all of their nodes are automatically considered as internal nodes. Note that to test if a node is inside, we only employ a subset of the faces of Ω^s. This is performed using the signed distance algorithm introduced in [23], which is also employed to compute the projection of a node onto a surface.

Lines 6–9 split all octants that do not comply with the 1-irregular property. Let us recall that the 1-irregular property means that neighbor octants have a difference in their RL no greater than 1. For this reason, when the loop ends, the transition patterns can be applied over the entire mesh at line 10. After this step, the mesh is finally topologically correct, meaning that no element in the mesh has a node inserted in an edge. In our visualization options, we display faces that belong to only one element in a different color. It is for this reason that some of the internal faces of Figure 15(a) are displayed differently. At the same time, it allows us to say that all these issues are overcome in Figure 15(b) by the use of transition patterns.

In terms of time efficiency, the search for the appropriate pattern is carried out very fast. We define a data structure that for each pattern will keep a key and the set of sub-elements for this pattern. A 1-irregular octant has a single vector with the eight nodes of the cube and mid-edges, mid-faces, and mid-hexahedron nodes (27 nodes in total). The key is actually a mask that is computed in terms of inserted nodes, regardless of the eight original nodes. Let us say that we have a pattern with nodes 11 and 13, which are mid-edge nodes. The mask for this pattern would be $2^{11} + 2^{13}$ in terms of bits; therefore, it would be the number 10240. Then, we insert this pattern with this key and their sub-elements in a map data structure. The main advantage of the map is that search, removal, and insertion operations have logarithmic complexity. Finally, if we want to search the pattern for a given configuration, we compute its mask as explained before and search for it in the map. Then, we will keep performing rotational operations and re-computing its mask until we find the pattern

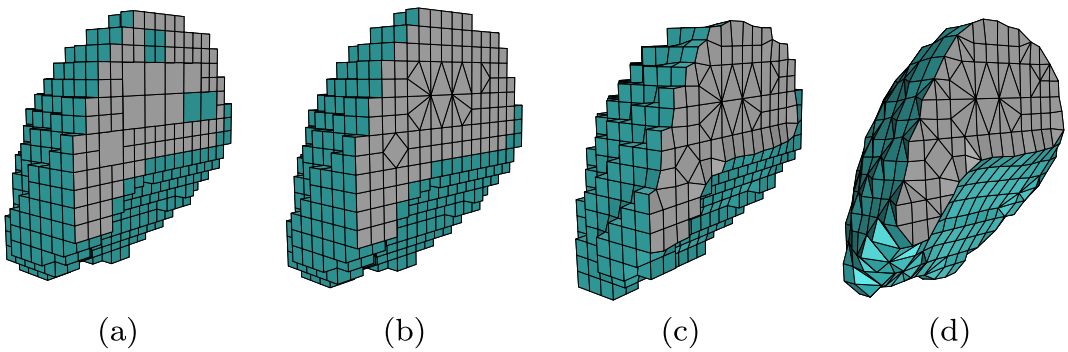


Figure 15. (a) The 1-irregular mesh, (b) using the transition patterns to achieve congruent topology, (c) projection onto Ω^s of internal nodes to avoid extremely flat elements, and (d) the use of surface patterns to better approximate the boundary of Ω^s.

DOI: 10.1002/cnm

(defined by its key). In other words, the search of a pattern has logarithmic complexity. In case you are not familiar with computational complexity, let us say that the example of the liver was entirely produced in less than a second with a MacBook Pro 2.7 GHz Intel Core i7.

It is very common to find surface elements close to the input's boundary. In several cases, the best solution will be to take them out of the mesh, causing a small deformation on neighbor elements. By doing this, we try to avoid quality issues like having flat elements at the end of the process. This is performed at line 11, where all the inside nodes of surface elements are pushed out onto Ω^s, if the distance between the current and the projected position is small. Once the displacement of all nodes is committed, the surface elements that no longer count with an inside node are removed from the mesh. This is shown in Figure 15(c).

It is now necessary to define what a small distance is. This is a parameter defined per node in terms of the average size of edges incident to it. In our experience, an arbitrary value of 30% of the average size produces acceptable quality results. If we consider the case of a grid type of mesh, this means that elements will not be flatter than 30% of the standard cube size, and also, that internal elements will not be artificially increased to an extra 30% size of the perfect element.

The aforementioned idea can be seen in Figure 16. First, the result is shown if the internal node P, which is close to the boundary, is not displaced (Figure 16(a)–(c)). The resulting element associated to octant number 1 is a poor quality triangle. On the other hand, if node P is first projected onto Ω^s, octant 1 must be removed as it does not count any longer with an inside node. The final result shown in Figure 16(f) presents better quality than the result of Figure 16(c).

Following the work presented in [1], line 12 of the algorithm invokes a process that will replace all hexahedral elements intersecting the surface of the input domain by mixed elements. These new elements tend to better represent curved domains, which is the case of anatomical structures. In the example of Figure 16, this process can be seen in Figure 16(b), where octants 1 and 4 have been replaced with mixed elements. It can also be seen in Figure 16(e), where the same procedure is performed for octant 2, replacing the square face by a triangle, octant 4, just leaving the square face as it is due to its particular configuration, and octant 5, replacing the square by two triangles. It is important to note that internal projected nodes are managed as outside nodes by all octants. Octant number 5 could stay as it was because as a quadrilateral face, it did not cause any topological inconsistency among neighbor elements. However, this cannot be extended to all cases. Therefore, it is safer to consider a unique state (inside or outside) for each node with respect to all of the octants.

In this article, we also introduce new surface patterns for the pyramid and the wedge (prism) because, due to the use of transition patterns, these type of elements may also be found at the surface of the resulting mesh. These new patterns were introduced in Section 4.2. Figure 15(d) shows the state of the liver example after the use of surface patterns.

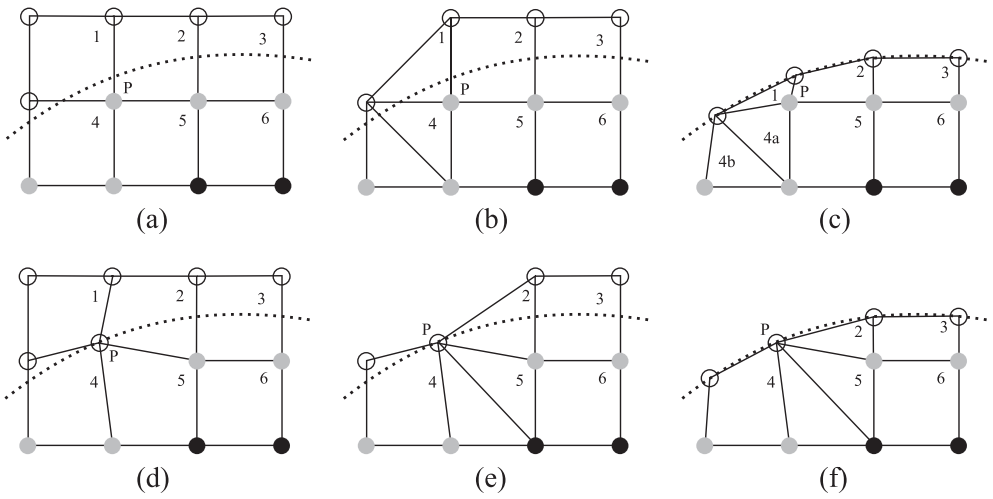


Figure 16. The use of surface patterns without (a)–(c) and with (d)–(f) internal node displacement.

DOI: 10.1002/cnm

Now, it is possible to project each outside node onto Ω^s. This corresponds to line 13 of the algorithm. This task is performed taking advantage of the Octree data structure. We first search all the octants that share the node to be projected. Among all the faces that are intersected by these octants, we search for the closest projection using, once more, the signed distance algorithm of [23], and this will be the final position for this node. The result of this process can be seen in Figure 16(f), and for the liver example, Figure 17(a) shows the state before projection and (b) after projection.

Both meshes have 3453 nodes and 6397 elements, all of them in a valid state. The 'surface patterns' mesh, shown in Figure 17(a), has 31 elements with a quality below 0.2, being the worst tetrahedron of 0.063. The mesh with projected nodes shown in Figure 17(b) has 41 elements below 0.2 and the worst is also a tetrahedron of 0.02. This tetrahedron is located at the surface in the region where transition patterns are employed. Note that to produce these results, we have not used any quality improvement technique nor a smoothing technique. The quality of our meshes will be discussed in Section 8.

This could be the last state of the mesh; however, this output clearly presents a staircase effect in some regions. This problem occurs when there is a drastic change in the normals of input neighbor faces. In other words, sharp features of Ω^s.

One step that helps to reduce the undesired staircase effect is to add some extra elements by some surface patterns that are normally found at this type of regions (sharp features). The idea is simple: if all the nodes that conform an element already belong to the mesh, we will keep this element if its quality is acceptable, because we know all of its nodes are currently on the surface Ω^s. The three patterns we consider to add extra elements are the ones shown in Figure 18. In this figure, it is clear that the white nodes, needed by the optional elements, are already in use by the other sub-elements of the pattern that will stay in the mesh. Therefore, it is worth to analyze if its quality is acceptable on behalf of the better surface representation.

The arbitrary threshold we use to add an element is a $J_{ENS} > 0.2$. We use this number because the original J_S considers hexahedra over this threshold to present acceptable quality. Moreover,

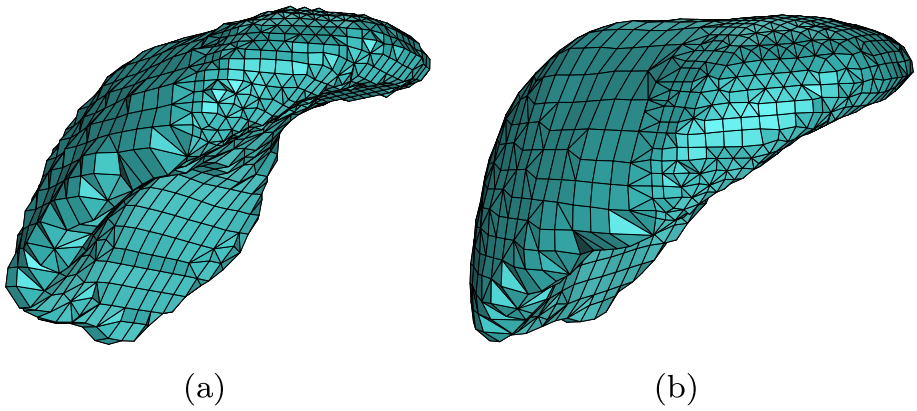


Figure 17. (a) The mesh after the use of surface patterns and (b) the mesh after node projection onto the surface.

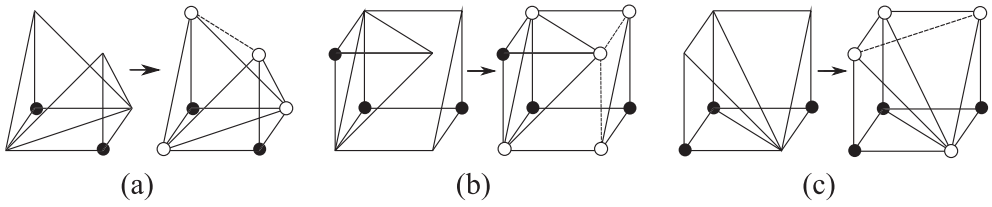


Figure 18. Surface patterns of [1] that include a new possible element: (a) pattern 2B, an extra tetrahedron; (b) pattern 3B, an extra pyramid; and (c) pattern 3C, an extra tetrahedron.

DOI: 10.1002/cnm

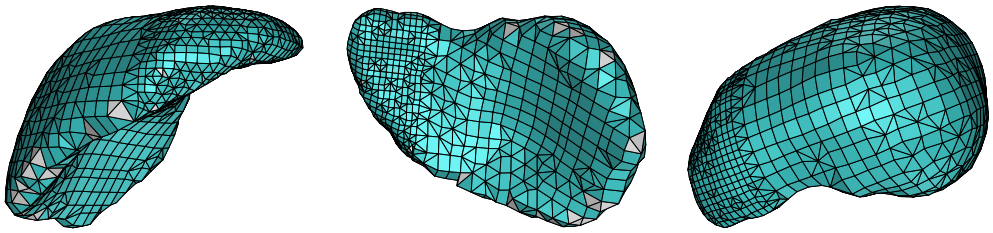


Figure 19. Different views of the output mesh: white faces belong to the optional elements that are finally kept in the mesh.

we have already detected elements with inferior quality; therefore, adding these elements will not decrease the quality of the mesh. Once more, we will discuss more about quality in Section 8.

The final step is to write the output volume mesh, which employs mixed elements for transition regions and at the surface. The rest of the elements will be hexahedra. The final output mesh can be seen in Figure 19.

Now, let us use the liver example to validate our new quality metrics with respect to tetrahedra. For the 2220 tetrahedra in the mesh, we have computed both the J_{ENS} and the ARG^{-1}. We can say that $J_{ENS} \in [0.02, 0.96]$ with an average of 0.5, while $\text{ARG}^{-1} \in [0.02, 0.99]$ with an average of 0.64. Knowing that a value closer to 0 indicates worst quality, the J_{ENS} is more restrictive in 2204 cases, meaning that with respect to ARG^{-1}, it penalizes more poor quality tetrahedra.

The greater difference in favor of J_{ENS} was 0.31 (more restrictive). While for the other 16 cases in which ARG^{-1} was more restrictive, the difference in favor of it was 0.14. We have to emphasize that on those 16 cases, the values of both metrics were superior to 0.4, meaning a good quality element from any point of view. Therefore, we can say that J_{ENS} is as good as ARG^{-1} to find poor quality tetrahedra.

Knowing that the behavior of J_{ENS} is acceptable for tetrahedra and hexahedra, and in the lack of other quality metric widely accepted for other types of elements, we consider that the J_{ENS} is validated to find poor quality elements and, moreover, is in most cases more restrictive than the ARG^{-1} and the J_S explained in Section 3.

7. RESULTS

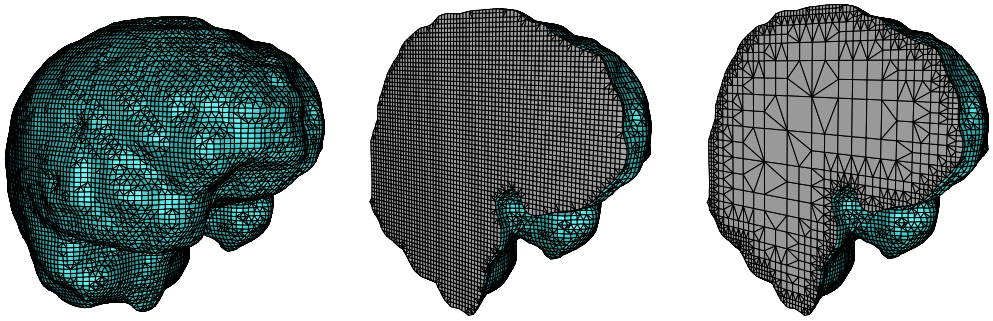
The first example that illustrates the results we obtained with the proposed meshing technique in this work is the problem known as 'brain shift'. The goal in this problem is to simulate, in real time, the mechanical deformation suffered by the brain during tumor resection surgery. As time goes by, during surgery, the location of the tumor changes with respect to original medical images, taken before skull opening. The idea is to track the position of the tumor in order to reduce eventual damage to the brain.

How the FE model is updated or other remarks regarding surgery is out of the scope of this work. However, we can say that in order to achieve this simulation, during surgery, it will be necessary to count with high precision in the region of the tumor, and less resolution elsewhere. In fact, it is this last point that allows optimization, as it will be shown with the next meshes.

In order to have a reference, Figure 20 shows two meshes: one where all octants are refined to the same level (Figure 20(b)) and second where only octants intersecting the surface are refined; the rest is meshed using the transition patterns (Figure 20(c)). The RL of both meshes is 6, meaning that the original octants were recursively split in eight new octants six times. For this reason, both meshes look the same at the surface; this can be seen in Figure 20(a).

One hypothetical scenario would be to have the tumor at the top of the brain; therefore, we would need to be very accurate at that zone in terms of the simulation. This is equivalent to having the smallest and better quality elements at the top of the brain, which would define our RoI in this particular case. If we do not count with the transition patterns, the only solution will be to produce a grid mesh, as shown in Figure 20(b). However, this adds several nodes out of the RoI that are

DOI: 10.1002/cnm



(a) (b) (c)

Figure 20. Two brain meshes that look the same on the surface: (a) but different on the inside, (b) regular internal elements, and (c) using transition patterns to reduce the quantity of nodes on the inside.

Table IV. Summary of node and element number, overall time in a MacBook Pro 2.7 GHz Intel Core i7, and quality distribution for all the meshes.

	Quantity		$t[s]$	J_{ENS}					
	Nodes	Elements		< 0.03	< 0.2	< 0.4	< 0.6	< 0.8	< 1
Liver	3453	6417	1	1	40	1521	1785	1420	1650
B-G	170959	199440	16	0	6	15822	13499	18963	151150
B-I	49431	119910	14	0	92	39872	42211	20576	17159
B-T	51126	75990	8	8	156	14044	13818	9942	38022
S.P.	74118	134781	16	2	343	33171	35073	27809	38383
F.B.	63147	139058	27	0	131	40261	48310	28469	21887

B-G, brain grid; B-I, brain irregular; B-T, brain top refinement; S.P., soft palate; F.B., foot bones. All the elements in all meshes present positive J_{ENS}.

Table V. J_{ENS} quality per element type.

J_{ENS}		Liver	B-G	B-I	B-T	S.P.	F.B.
$Hex.$	min.	0.589	0.4	0.4	0.488	0.55	0.37
	max.	1	1	1	1	1	1
	avg.	0.976	0.99	0.966	0.995	0.98	0.959
	qty.	1316	147516	11779	36026	32605	14900
$Pri.$	min.	0.334	0.467	0.468	0.324	0.336	0.07
	max.	0.987	0.975	0.975	0.975	0.987	0.99
	avg.	0.694	0.674	0.701	0.683	0.435	0.7
	qty.	862	11419	12101	5435	41817	16956
$Pyr.$	min.	0.06	0.141	0.085	0.011	0.008	0.046
	max.	1	0.982	1	0.968	1	1
	avg.	0.434	0.456	0.41	0.41	0.435	0.427
	qty.	2019	20349	47741	17281	41817	52176
$Tet.$	min.	0.02	0.129	0.125	0.003	0.028	0.11
	max.	0.964	0.992	0.992	0.991	1	1
	avg.	0.5	0.539	0.48	0.509	0.5	0.49
	qty.	2220	20156	48289	17248	43526	55026

B-G, brain grid; B-I, brain irregular; B-T, brain top refinement; S.P., soft palate; F.B., foot bones.

DOI: 10.1002/cnm

unnecessary due to fact that almost no displacement will occur at those regions. The grid mesh has a total of 170,959 nodes and 199,440 elements. The other extreme would be to only refine elements at the surface, in order to have an idea of what we can save in terms of node quantity, while achieving the same level of representation. This mesh has 49,431 nodes and 119,910 elements. The statistics of all the meshes presented in this work will be summarized in Tables IV and V. However it is important to highlight that the irregular mesh (only surface refinement) employs only a 29% of the nodes of the regular mesh for the brain.

To fulfill the constraints of the given problem, we will demand our meshing technique to produce a mesh where the minimum RL for an octant will be level 4. Any element intersecting the surface must have at least an RL 5, and most important, all the elements at the top of the brain (defined by a given coordinate) must have an RL 6. The resulting mesh can be seen in Figure 21. This mesh has 51,126 nodes and 75,990 elements, that is, it saves 70% of the nodes employed by the 'grid' mesh as shown in Figure 20(b).

It is important to recall that the focus of this problem was to count with as much as possible high-quality elements at the top of the brain and leave the rest with acceptable quality and, at the same time, with as few nodes as possible to enable fast computation in the context of a real-time simulation. For this reason, the surface of the middle–bottom section of the brain was refined to level 5, which allows to drastically reduce the quantity of nodes and obtain a similar result (in nodes quantity) to the mesh generated with surface refinement 6, already shown in Figure 20(c).

A 'soft palate' mesh can be seen in Figure 22. Here, the refinement of the top region and some region of the bottom was refined to level 7. Other regions at the surface have an RL 6 and the rest is fulfilled with transition patterns.

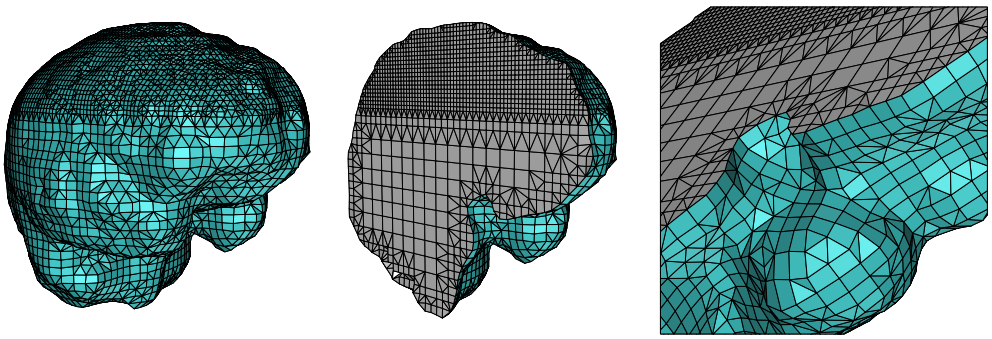


Figure 21. Brain mesh with top refinement at level 6, the rest of the surface at level 5, and inside elements to level 4, except for transition octants: (a) surface view, (b) sagittal cut, and (c) zoom to same sagittal cut.

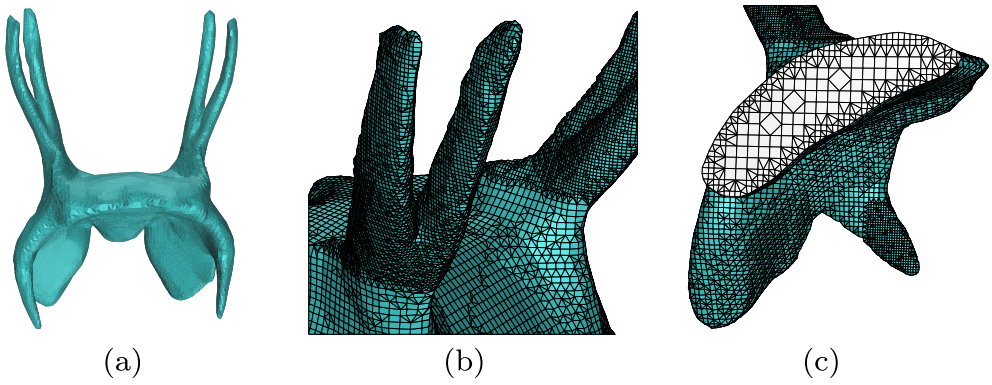


Figure 22. Soft palate mesh: refinement at top and some regions at the bottom with level 7 and the rest of the surface with level 6: (a) surface view, (b) a zoom to the top region, and (c) sagittal cut.

DOI: 10.1002/cnm

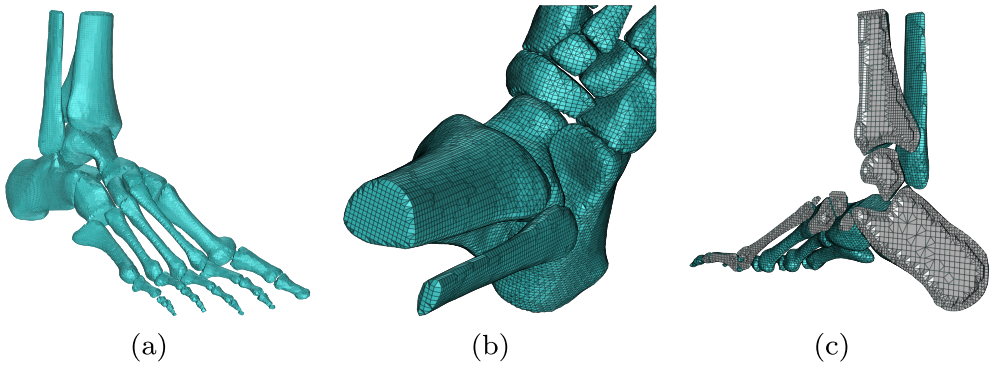


Figure 23. Foot bones mesh: surface refinement of level 6 and fulfilled with transition patterns: (a) surface view, (b) a zoom viewed from the top, and (c) sagittal cut.

Our final example is the model of the bones of a foot. Here, we refined surface elements to level 6 and managed the rest with transition patterns. This mesh counts 63,147 nodes and 139,058 elements. Here, despite the fact that this geometry is more complex to mesh than the others, we obtain better quality results because the surface RL is the same for the entire domain, meaning that no transition pattern is employed at the surface. This mesh can be seen in Figure 23.

With respect to the time our algorithm takes to generate a mesh, there is no direct conclusion as it drastically depends on the input surface mesh. A mesh with more triangles takes more time because the inside/outside test and the projection must be computed using more faces. It is for this reason that in the brain example, the mesh with greater refinement at the top takes less time; the mentioned tests must be performed by less octants over the same quantity of faces for the middle-bottom region.

8. CONCLUSION, DISCUSSION, AND FUTURE WORK

In this work, we have introduced a new meshing technique based on the Octree structure. This meshing technique allows the coexistence of coarse and fine regions in the same mesh by a set of transition patterns introduced in this work. We have implemented the total of 325 patterns that thanks to rotation operations allow us to manage all possible cases. The focus of these patterns was to introduce as few nodes as possible, with the goal to be used under real-time simulations. In contrast with other works [4, 6–9, 11], we will only add one node for some patterns that otherwise would not be split in simple elements. We have found 65 patterns that need the insertion of an octant's central node.

The main goal was to have more control on the quantity of nodes by the implementation of all the patterns. Several other works implement a subset of the patterns, and if for a given configuration, a pattern does not exist; they split the octant. Each time an octant is split, it adds up to 21 new nodes in the case of splitting in eight new cubes, and in the case of splitting in 27 new cubes (transition with hexahedra only), it adds 56 new nodes. Therefore, it is easy to expand the regions of high refinement with several more nodes and elements than the minimal quantity.

Another aspect to study was the quality of the elements that are produced with these transition patterns. One main problem was to establish which metric to use when the mesh has mixed elements. In the lack of such a quality metric, we have proposed in this article a new metric that we call element normalized scaled Jacobian J_{ENS} based on the widely accepted scaled Jacobian J_S for hexahedral meshes. In this work, we adapted the J_S to measure the quality of tetrahedra, and we contrasted this new metric to a widely used metric for measuring tetrahedron quality, the aspect ratio gamma ARG^{-1}. The values we obtained were similar when measuring poor quality tetrahedra. Moreover, the few cases we detected where the J_{ENS} was more permissive than the ARG^{-1} were cases of acceptable quality elements (>0.4).

Once the J_{ENS} was established for hexahedral and tetrahedral elements, it was easy to extend it to the pyramid and prism, which are the other element types employed by our technique. If other type of element was employed, we would need to create a version of it where all its faces should

DOI: 10.1002/cnm

be equilateral, then measure the J_S to find its constant and employ the same formula introduced in Section 3.

Now, one of the main advantages of having a metric that serves for different types of elements is to be able to use it in repairing methods. Several of these algorithms use a quality metric to determine over which direction the node should be displaced in order to increase the quality of the elements attached to it. If we use different functions to measure the quality of each type of element, it is sometimes not clear which direction will increase the quality of the system. This should be analyzed in more details in further works.

With the J_{ENS} established as a valid quality metric, we performed a study for the elements in the transition patterns. This reveals that some of the elements in the patterns present very poor quality from the beginning. This was not a surprise because the goal of the patterns, as mentioned before, was to decrease the number of nodes in the mesh. And in doing so, we did not focused on quality issues. Now, we can employ a mesh generated with our technique and fix it as the one that has the minimum node quantity. Then, by adding nodes increases the quality and let the user decide the right trade off between element quality and node quantity.

Several solutions must be analyzed. For instance, not allowing poor quality transition patterns at the surface is because they will decrease even more its quality once some of its nodes are projected onto the surface. If a poor quality pattern is needed, then we refine the octant instead, as the other techniques do. In the worst scenario, this will end up in regular refinement at the surface. Another option is to consider splitting the poor quality pattern using not the center of the octant but other position that increases overall quality in the octant. Moreover, do not use only one new node, maybe add two internal nodes to remesh the octant and comply with face patterns shared with neighbors. As it can be seen, there are options to overcome the problems given by poor quality patterns; however, it is important to remark that, now that we have implemented the whole set of patterns, we know which ones are of poor quality and should be avoided.

It is impossible to ensure that our meshing technique will never produce poor quality elements nor invalid elements. However, we have shown through examples that even for complex geometries, we produce few poor quality elements. Moreover, we have not used any quality filter, like a Laplacian filter, to increase the quality. Therefore, an option to overcome quality issues would also be to use repairing methods like [27] or quality enhancers like [28]. We analyzed the quality of the worst mesh introduced in this paper with ANSYS. This is the mesh of the 'brain with top refinement' (B-T), shown in Figure 21. The mesh was suitable for an FE simulation. There was no error found, and only 62 out 75,990 (0.08%) of the elements produced a warning in terms of quality.

In this work, we have also extended the surface patterns to all type of elements, allowing a transition pattern to be used at the surface (even if this depicts quality issues). Combining transition with surface patterns will allow to increase the precision of the simulation in the RoI, or decrease computational time of the simulation (it can be seen both ways), enabling this technique for the use in real-time applications.

In order to improve the approximation of the input boundary, we are already working on feature detection in order to adapt our surface and transition patterns to capture them. This will avoid the undesired staircase effect that may be produced at low RLs.

Finally, we would like to mention that this meshing technique has already been employed in some health studies like [29] and [30]; therefore, it has already shown to be a good alternative for meshing problems. Moreover, it will be freely available at a future version of CamiTK[‡] and ArtiSynth[§].

ACKNOWLEDGEMENTS

Several people have contributed to this work with ideas, discussion, and help. Special thanks to Nancy Hitschfeld, Yohan Payan, Marek Bucki, Vincent Luboz, and Jorge Sepúlveda. The authors would like to thank project: Fondecyt de Iniciación 11121601.

Contract/grant sponsor: FONDECYT Iniciación; contract/grant number: 11121601.

[‡]http://camitk.imag.fr/
[§]http://www.artisynth.org/

DOI: 10.1002/cnm

REFERENCES

1. Lobos C. A set of mixed-elements patterns for domain boundary approximation in hexahedral meshes. *Studies in Health Technology and Informatics* 2013; **184**:268–272. Proceedings of MMVR20.
2. Yerry MA, Shephard MS. Automatic three-dimensional mesh generation by the modified-Octree technique. *International Journal for Numerical Methods in Engineering* 1984; **20**(11):1965–1990.
3. Shephard M, Georges M. Automatic three-dimensional mesh generation by the finite octree technique. *International Journal for Numerical Methods in Engineering* 1991; **32**:709–749.
4. Schneiders R. Refining quadrilateral and hexahedral element meshes. *Proceedings of the Fifth International Conference on Numerical Grid Generation in Computational Field Simulations*, NSF Engineering Research Center for Computational Field Simulation, College of Engineering, Mississippi State University, Mississippi State, MS 39762, USA, 1996April; 679–688.
5. Schneiders R, Schindler R, Weiler F. Octree-based generation of hexahedral element meshes. *In Proceedings of 5th International Meshing Roundtable*, Sandia National Laboratories, Pittsburgh, PA, USA, 1996; 205–215.
6. Zhang Y, Bajaj C. Adaptive and quality quadrilateral/hexahedral meshing from volumetric data. *Computer Methods in Applied Mechanics and Engineering* 2006; **195**(9–12):942–960.
7. Zhang H, Zhao G. Adaptive hexahedral mesh generation based on local domain curvature and thickness using a modified grid-based method. *Finite Element Analysis and Design* 2007; **43**(9):691–704.
8. Ito Y, Shih A, Soni B. Octree-based reasonable-quality hexahedral mesh generation using a new set of refinement templates. *International Journal for Numerical Methods in Engineering* 2009; **77**(13):1809–1833.
9. Ebeida MS., Patney A, Owens JD, Mestreau E. Isotropic conforming refinement of quadrilateral and hexahedral meshes using two-refinement templates. *International Journal for Numerical Methods in Engineering* 20119; **88**(10):974–985.
10. Conti P, Hitschfeld-Kahler N, Fichtner W. Omega–an octree-based mixed element grid allocator for the simulation of complex 3-D device structures. *IEEE Transactions. on CAD of Integrated Circuits and Systems* 1991; **10**(10): 1231–1241.
11. Nicolas G, Fouquet T. Adaptive mesh refinement for conformal hexahedral meshes. *Finite Elements in Analysis and Design* 2013; **67**(0):1–12.
12. Finkel R, Bentley JL. Quad trees: A data structure for retrieval on composite keys. *Acta Informatica* 1974; **4**(1): 1–9.First article mentioning the quadtree data structure
13. Delaunay B. Sur la sphere vide. *Izvestia Akademii Nauk SSSR, Otdelenie Matematicheskikh i Estestvennykh Nauk* 1934; **7**:793–800.
14. Knupp P. Achieving finite element mesh quality via optimization of the jacobian matrix norm and associated quantities. part ii – a framework for volume mesh optimization and the condition number of the jacobian matrix. *International Journal for Numerical Methods in Engineering* 2000; **48**:1165–1185.
15. Knupp PM. A method for hexahedral mesh shape optimization. *International Journal for Numerical Methods in Engineering* 2003; **58**(2):319–332.
16. Owen SJ, Sorensen MC, Staten ML. Parallel hex meshing from volume fractions. *In Proceedings of 20th International Meshing Roundtable*, Springer-Verlag: Sandia National Laboratories, Paris, France, 2011October; 161–178.
17. Dudley CR, Owen SJ. Degenerate hex elements. *Procedia Engineering* 2014; **82**(0):301–312. 23rd International Meshing Roundtable (IMR23).
18. Zhang Y, Hughes T, Bajaj C. An automatic 3d mesh generation method for domains with multiple materials. *Computer Methods in Applied Mechanics and Engineering* 2010; **199**(5–8):405–415.
19. Qian J, Zhang Y. Sharp feature preservation in octree-based hexahedral mesh generation for cad assembly models. *Proceedings of the 19th International Meshing Roundtable, IMR 2010*, Springer-Verlag, Chattanooga, TN, USA, 2010; 243–262.
20. Parthasarathy V, Graichen C, Hathaway A. A comparison of tetrahedron quality measures. *Finite Elements in Analysis and Design* 1993; **15**:255–261.
21. Lobos C. Towards a unified measurement of quality for mixed-elements. *Technical Report 2015/01,* Departamento de Informática, UTFSM, 2015. (Available from: http://www.inf.utfsm.cl/~clobos/tech.html.) [Accessed on 30 May 2015.]
22. González E, Lobos C. A set of mixed-element transition patterns for adaptive 3d meshing. *Technical Report 2014/01,* Departamento de Informática, UTFSM, 2014. (Available from: http://www.inf.utfsm.cl/~clobos/tech.html.) [Accessed on 30 May 2015.]
23. Baerentzen J.A, Aanaes H. Signed distance computation using the angle weighted pseudonormal. *Visualization and Computer Graphics, IEEE Transactions on* 2005; **11**(3):243–253.
24. Wilf Herbert S. *Generatingfunctionology*. Academic Press, Inc., 1994.
25. Lobos C, Rojas-Moraleda R. From segmented medical images to surface and volume meshes, using existing tools and algorithms. *Proceedings of ADMOS*, International Center for Numerical Methods in Engineering (CIMNE), Lisbon, Portugal, 2013; 436–447.
26. Foley JD. *Computer Graphics: Principles and Practice*. Addison–Wesley Professional, 1996.
27. Bucki M, Lobos C, Payan Y. A fast and robust patient specific finite element mesh registration technique: Application to 60 clinical cases. *Medical Image Analysis* 2010; **14**(3):303–317.

DOI: 10.1002/cnm

28. Freitag L. On combining Laplacian and optimization-based mesh smoothing techniques. *In Trends in Unstructured Mesh Generation*, Vol. 220, ASME Applied Mechanics Division, Amer Society of Mechanical, Dallas, Texas, USA, 1997; 37–44.
29. Bucki M, Luboz V, Lobos C, Vuillerme N, Cannard F, Diot B, Payan Y. Patient-specific finite element model of the buttocks for pressure ulcer prevention - linear versus non-linear modelling. *Computer Methods in Biomechanics and Biomedical Engineering* 2012; **15**(sup1):38–40.
30. Rohan P-Y, Lobos C, Nazari MA, Perrier P, Payan Y. Finite element modelling of nearly incompressible materials and volumetric locking: a case study. *Computer Methods in Biomechanics and Biomedical Engineering* 2014; **17**(sup1):192–193. PMID: 25074229.