The receding front method applied to hexahedral mesh generation of exterior domains

Eloi Ruiz-Gironés · Xevi Roca · Josep Sarrate

Received: date / Accepted: date

Abstract Two of the most successful methods to generate unstructured hexahedral meshes are the grid-based methods and the advancing front methods. On the one hand, the grid-based methods generate high-quality hexahedra in the inner part of the domain using an inside-outside approach. On the other hand, advancing front methods generate high-quality hexahedra near the boundary using an outside-inside approach. To combine the advantages of both methodologies, we extend the receding front method: an inside-outside mesh generation approach by means of a reversed advancing front. We apply this approach to generate unstructured hexahedral meshes of exterior domains. To reproduce the shape of the boundaries, we first pre-compute the mesh fronts by combining two solutions of the Eikonal equation on a tetrahedral reference mesh. Then, to generate high-quality elements, we expand the quadrilateral surface mesh of the inner body towards the unmeshed external boundary using the pre-computed fronts as a guide.

Keywords Mesh generation · unstructured hexahedra · eikonal equation

1 Introduction

During the last two decades several general-purpose algorithms for fully automatic hexahedral mesh generation have been proposed, see [1–6] for a survey. However, none of the existent algorithms is robust, automatic and generates high-quality meshes for any initial geometry. There are two families of methods that almost fulfill all these requirements, the *grid-based* and the *advancing front* methods. In fact, these approaches are the most successful methodologies to obtain a general-purpose hex-meshing algorithm. Furthermore, the grid-based and advancing front methods have advantages and disadvantages that complement each other. Thus, we can consider how to obtain a hexahedral meshing approach that presents only the advantages, and avoids the disadvantages, of these two methods.

On the one hand, the standard grid-based methods [7–11] are the only family of hexahedral mesh generation algorithms that are robust and fully automatic. In addition, they generate high-quality meshes in the inner part of the mesh. These advantages are possible because the mesh is generated from inside to outside. However, the grid-based methods generate low-quality hexahedra near the boundary and the final mesh depends on the spatial orientation of the domain. These drawbacks appear because the inner mesh does not have layers of hexahedra that progressively adapt to the boundary shape of the domain.

On the other hand, the advancing front methods [12–15] generate high-quality meshes near the boundary (*boundary sensitive*) that do not depend on the orientation of the object (*orientation insensitive*), see details on hex-meshing requirements in [2]. This is possible because the elements are generated layer by layer following the shape of the boundary surface. However, the advancing front methods are less robust and automatic. When the fronts are advanced, from the boundary to the inner part, they collide and can delimit complex voids. Specifically, if the advancing front method starts with a prescribed quadrilateral mesh of the boundary (*constrained* approach) [12] the resulting void is, in general terms, over-constrained and cannot be meshed. On the con-

This work was partially sponsored by the Spanish *Ministerio de Ciencia e Innovación* under grants DPI2007-62395, BIA2007-66965 and CGL2008-06003-C03-02/CLI and by Universitat Politècnica de Catalunya (UPC)

E. Ruiz-Gironés · X. Roca · J. Sarrate
Laboratori de Càlcul Numèric (LaCàN),
Departament de Matemàtica Aplicada III,
Universitat Politècnica de Catalunya,
Jordi Girona 1-3, E–08034 Barcelona, Spain
{eloi.ruiz,xevi.roca,jose.sarrate}@upc.edu

trary, the versions of the advancing front method that start without a prescribed mesh of the boundary (*unconstrained* approach) [13–15] can always generate a hexahedral mesh for the void. However, the quality of the mesh of the inner void is not guaranteed because it results from splitting each tetrahedron of a tetrahedral mesh into four hexahedra. These disadvantages at the inner part are caused because the elements are generated from outside to inside. Note that there are also constrained methods that directly transform a tetrahedral mesh into a hex-dominant mesh [16,17], and other unconstrained methods that using a reference tetrahedral mesh generate full-hex meshes [18–20].

In summary, by generating elements from inside to outside one can avoid the front collisions that lead to unmeshed voids or low-quality inner meshes. Moreover, by generating the elements using fronts (layers of elements) one can obtain meshes that reproduce properly the shape of the domain boundary. For this reason, our long-term goal is to combine the advantages of these methods into a general-purpose hexahedral mesh generation algorithm. In its general form the proposed method is composed by the following four steps:

(i) Extract an approximation of the medial axis of the volume.
(ii) Generate an inner hexahedral mesh (*seed*) that follows the medial axis approximation.
(iii) Generate level sets (fronts) between the hexahedral seed and the boundary of the volume.
(iv) Generate layers of elements from the inside to outside using the level sets as a guide.

We refer to this method as the *receding front method*. That is, a reversed advancing front method. In this work and in the previous ones [21,22] we apply the receding front method to generate unstructured hexahedral elements for exterior domain problems. Note that for these specific problems we do not need to obtain the medial axis and to generate hexahedral elements that follow it (first and second steps of the receding front method). In fact, the initial seed is an unstructured quadrilateral mesh over the surface of the inner object. Therefore, the main contributions of this work are focused on the third and fourth steps. That is:

(i) **To pre-compute the fronts (or layers of hexahedra) by solving a non-linear PDE**. We combine two solutions of the Eikonal equation to pre-compute the fronts. One solution determines the distance from the inner boundary and the other solution determines the distance from the outer boundary. The level sets of the combined distance field smoothly adapt to both boundaries. For this reason, the final mesh smoothly adapts from the inner boundary to the outer boundary.
(ii) **To generate layers of elements from inside to outside**. We expand a quadrilateral surface mesh defined on the inner body towards the outer boundary using the level sets as a guide. In this way, we avoid the collision of meshing fronts in the inner part. In order to generate the hexahedral elements, we propose to use a set of advancing templates that adapts the mesh front to the geometry features. In order to maintain the prescribed element size, we also propose to use a local refining process.

Note that combining two solutions of the Eikonal equation we have: (i) a global description in the interior of the geometry of its shape; and (ii) a not expensive procedure to pre-compute an approximate location of the layers of hexahedra (fronts). Using this information we insert a set of templates to advance from one level to the next one. The new nodes lay on the pre-computed fronts.

2 Related work

This work is clearly related to the grid-based and advancing front methods. However, the standard grid-based methods do not generate layers of hexahedra from inside to outside that smoothly adapt to the boundary of the domain. In addition, the advancing front methods do not start to generate layers of hexahedra from the inner part of the domain. Hence, the proposed approach is different to both methodologies. Furthermore, we propose to pre-compute the fronts by solving the Eikonal equation. It is important to point out that there are other mesh generation works that use the Eikonal equation. In his seminal work, Sethian proposes a method to advance structured meshes by solving the Eikonal equation [23]. Another front propagation method based on the Eikonal equation is presented in [24]. In [25,26], the authors show how to obtain the *medial axis transform* (MAT) by means of the Eikonal equation. Nevertheless, this is the first work where two solutions of the Eikonal equation are combined to pre-compute the fronts and obtain an unstructured hexahedral mesh.

Other techniques to guide the layering and alignment of the elements have been proposed in the literature. A geometrical guide is provided by the medial axis, which is used to align and layer the elements according to the shape of the domain [27–32]. Instead of using the geometrical description of the medial axis as a guide, we propose to use the level sets of a scalar field. Thus, we avoid to compute the medial axis and we only need to obtain two rough approximations of the solution of two Eikonal equation problems. Note that using a field as a guide has been previously proposed for mesh generation in [33,34]. Specifically, in these works a tensor field is used to control the alignment of a hex-dominant mesh in the three possible directions. Here we propose to use a scalar field to control the alignment of an all-hex mesh just in the advance direction. The alignment of

the other two directions is determined by the seed mesh and the proposed advancing templates.

The main drawback of the standard grid-based methods is the quality of the elements near the boundary. To alleviate it, the modified grid-based methods [35–38] insert a layer of hexahedra that reproduce the shape of the boundary. In these methods only the layers at the boundary can be unstructured and adapted to the shape of the geometry, while in our approach all the layers can be unstructured and adapted to the shape of the boundary.

Several approaches to advance from one constrained level to the next one have been proposed in literature. For instance, the plastering method [12] controls the number of elements in the new layer by using seams (decrease) and wedges (increase). In our approach this control is performed by the proposed templates (composed by several hexahedra). Specifically, we introduce a new set of templates that allow to obtain quality meshes around semi-vertex and semi-edge features. The whisker weaving approaches [39,40] also proposes how to advance from one level to the next one. However, in these approaches the advance is performed in the mesh dual instead of the mesh primal.

In our advancing process, the new layer of unstructured hexahedra only depends on the mesh of the previous level. That is, according to the geometry and topology of the previous level the algorithm automatically selects the proper hexahedral templates to generate the new layer. Then, if the new elements do not respect the prescribed element size an additional refinement step is performed. The refining templates are edge-based and several of them differ from the standard node-based templates [10]. Note that with the proposed techniques, we do not modify the elements on the previous level. In this sense, we could use general-purpose techniques to modify the current hexahedral mesh [41–43]. However, with our template based approach (specific-purpose) we explicitly determine the desired mesh. In addition, using these templates it is straightforward to avoid to modify the elements of the previous level.

3 2D Motivation

To illustrate and clarify the basis of the receding front method for a 3D geometry, we consider first a 2D example. Specifically, we present a smooth domain that will be meshed using quadrilateral (hexahedral) elements. With the help of this domain we first review the main advantages and disadvantages of the grid-based and advancing front methods. Finally, we outline the proposed receding front method, which combines the advantages of both methods.

Given a domain, the grid-based methods typically generate a quadrilateral (hexahedral) mesh in the inner part of the domain, Figure 1(a). Then, the remaining void between the inner mesh and the boundary has to be meshed, Figure 1(b). To this end, several new nodes are created on the boundary. These nodes are connected with the quadrilateral elements of the boundary of the inner mesh to form the last layer of hexahedra, Figure 1(c). The boundary of the inner mesh is not adapted to the boundary of the geometry and, for this reason, low-quality hexahedra close to the boundary may appear. However, this approach is robust and can be applied to general geometries to obtain meshes with high-quality elements in the interior of the geometry.

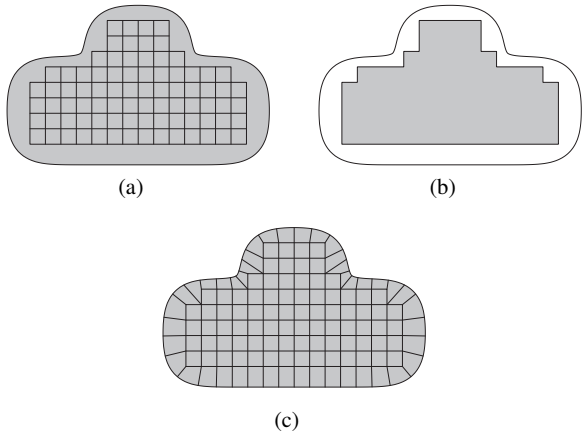


Fig. 1 Several steps of a grid-based method: (a) inner mesh; (b) void between boundary and inner mesh; and (c) final mesh.

In summary, grid-based methods generate high-quality elements in the interior of the volume and may generate low-quality elements near the boundary. These behavior appears because the mesh is generated from inside to outside.

Advancing front methods generate layers of elements (fronts) that start at the domain boundary and, layer by layer, advance towards the inner part of the domain. At the last step, several elements that connect the fronts close the remaining void. There are two families of advancing front methods: the constrained [12] and the unconstrained approaches [13–15].

The constrained approach generates a first layer of elements, Figure 2(a), that matches with a prescribed mesh of the boundary. Then, several layers of elements are generated by merging and matching the elements that are in front of the last layer, Figure 2(b). Inner voids defined by the fronts can be complex to mesh. In addition, since the process starts with a prescribed boundary mesh, the inner voids can be over-constrained, which leads to low-quality elements or inner voids that cannot be meshed, as illustrated in Figure 2(c).

The unconstrained approach relaxes the hex-meshing problem by considering that the domain boundary is not previously meshed. The meshing process starts at the boundary and provides a decomposition of the domain in several layers, Figure 3(a). The process stops when the inner void can

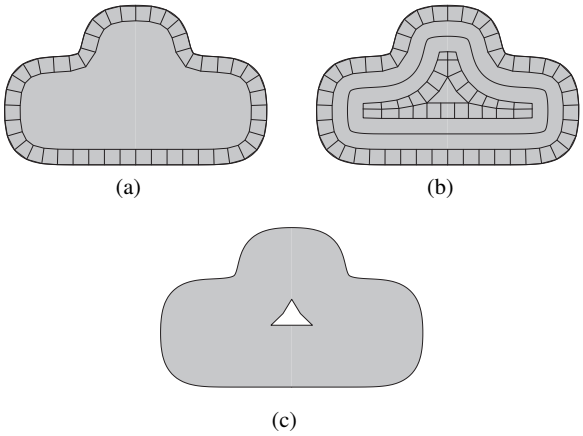


Fig. 2 Several steps of a constrained advancing front method: (a) first front; (b) last front and contours of the previous fronts; and (c) unmeshed void.

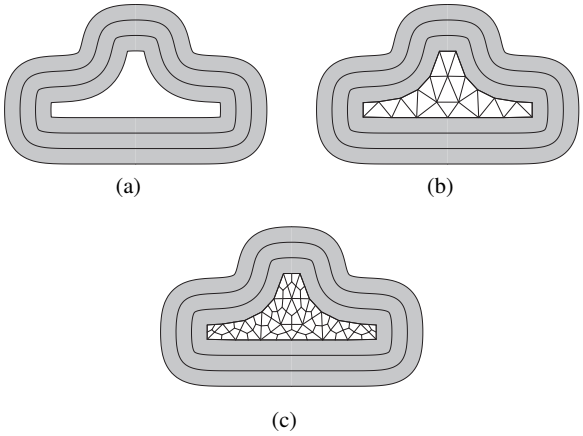


Fig. 3 Several steps of an unconstrained advancing front method: (a) fronts and final void; (b) simplicial mesh of the void; and (c) splitting simplicial mesh.

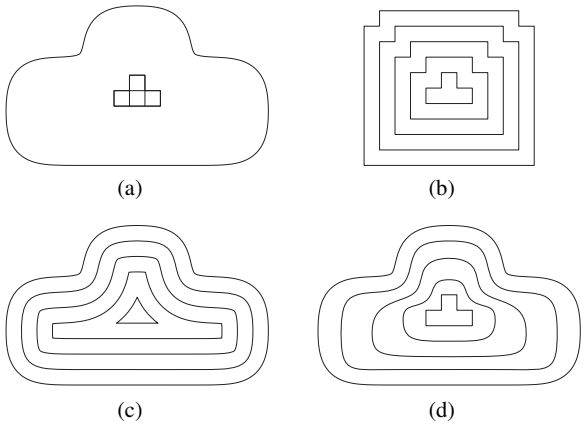


Fig. 4 Pre-computing the fronts: (a) outer boundary and inner seed; (b) level sets from inside to outside; (c) level sets from outside to inside; and (d) combining inside-to-outside with outside-to-inside level sets.

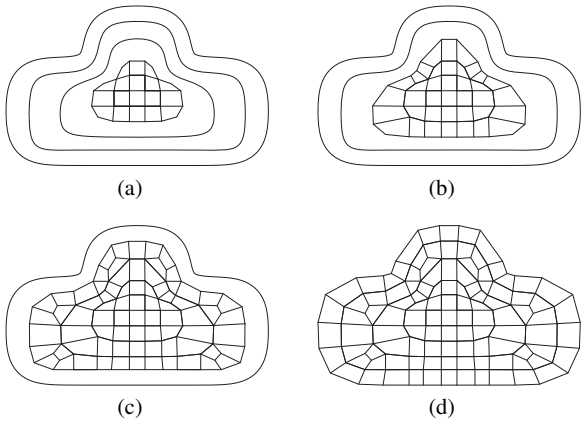


Fig. 5 Layers of elements for the receding front method: (a) first layer; (b) second layer; (c) third layer; and (d) final layer.

be discretized with a hex-meshing primitive. Since the inner void results from successive offsets of the boundary, it can be as much difficult to hex-mesh as the initial domain. Thus, there are configurations where the inner void cannot be meshed with a high-quality hex-meshing primitive. However, it is possible to generate a hexahedral mesh from a simplicial mesh of the unrecognized inner void, Figure 3(b). The simplicial mesh is split in quadrilateral (hexahedral) elements, Figure 3(c). Then, the boundary of this inner mesh is propagated through the layers towards the boundary of the domain. The quality of the inner elements is not guaranteed because they are originated by a simplicial mesh. Moreover, since the boundary of the inner mesh is propagated through the domain towards its boundary, the inner mesh determines the structure and the quality of the mesh at boundary curves (surfaces). It is important to point out that the element quality close to the boundary features is ensured because it is a front-based approach. As for all mesh generation algorithms, low-quality elements may appear close to very sharp dihedral angles. In addition, this approach is fully automatic and can provide high-quality meshes for a wide range of geometries.

In summary, advancing front methods generate high-quality elements near the boundary of the volume and may generate low-quality elements in the inner part. This behavior appears because the mesh is generated from outside to inside.

To combine the advantages of both the grid-based and the advancing front methods, herein we propose the receding front method. This methodology requires an initial mesh (seed) of the inner part of the domain, Figure 4(a). Note that for the specific case of meshing the exterior domain of a given body, the initial mesh is a quadrilateral mesh of the body surface. This allows to decouple the problem of generating the inner seed from the front generation process. The fronts that determine the layers of elements can be pre-computed. Specifically, first we generate offset levels of the shape of the inner seed towards the boundary, Figure

4(b). Second, we compute the offset levels of the shape of the outer boundary inwards, Figure 4(c). These offset levels are obtained as the level sets of two solutions of the Eikonal equation, see Section 4.1. One solution is related to the boundary of the inner seed and the other one to the outer boundary. To compute these solutions we use an edge-based solver on a triangular (tetrahedral) mesh, see [44]. Third, we combine both solutions to obtain a function that reproduces the shape of the inner body in the inner part, and the shape of the boundary close to the outer part, Figure 4(d). Finally, the mesh fronts are obtained as the level sets of this function. In Section 4.2 these fronts are used as a guide to generate layers of elements starting from the inner seed and advancing towards the outer boundary, Figure 5. To this end, a set of templates determines how to offset the previous layer of elements to the new front. Moreover, we have to consider a set of refinement rules that ensures that the element size is not exceeded. The resulting procedure generates layers of elements that progressively morph from the shape of the inner boundary to the shape of the outer boundary. In addition, starting from the inner part we can avoid over-constrained or complex inner voids.

4 The Receding Front Method

The receding front method is decomposed into two steps. First, we pre-compute a set of fronts between the inner and the outer boundaries. Second, we generate layers of elements from inside to outside by expanding the quadrilateral mesh of the inner boundary towards the unmeshed outer boundary according to the pre-computed fronts.

4.1 Pre-Computing the Fronts

Given a domain $\Omega \subset \mathbb{R}^n$, the Eikonal equation is the following non-linear partial differential equation

$$\begin{cases} \|\nabla d\| &= f \text{ in } \Omega \\ d|_{\mathcal{U} \subset \{\Omega \cup \partial\Omega\}} &= 0, \end{cases} \tag{1}$$

where f is a known function, $\|\cdot\|$ is the Euclidean norm and $\mathcal{U}$ is a sub-set of the closure of Ω. For $f = 1$ the solution d is the distance from $\mathcal{U}$. For $f = \frac{1}{h(x)}$ the level sets of the solution d follow the size field $h(x)$ defined for $x \in \Omega$. In this work we consider $f = 1$. In our implementation, the Eikonal equation is solved on a tetrahedral mesh by means of an edge-based solver. This solver is a modification of the solver presented in [44].

Algorithm 1 presents the proposed procedure to compute the solution of the Eikonal equation (1). Given a mesh and a set of nodes that belongs to $\mathcal{U}$, initialize the values of the solution at zero for these nodes and to infinity for all other nodes. In addition, as the values of the solution are initialized, the nodes are inserted in a min-heap, Lines 3–10. A min-heap is a data structure in which the values are stored in a descending order. The nodes are sorted in the min-heap according to its current value of the solution. That is, the nodes with the smaller values come before than the nodes with greater values. The main idea of the algorithm is that the node with the smaller value of the solution contains a correct value. Note that the node with the smallest value is the first node of the min-heap. Such node is removed from the min-heap (Line 13) and the value of the solution of its adjacent nodes is updated according to the Eikonal equation. Let n_0 and n_e be the node with smallest value and an adjacent node through an edge e, respectively. According to Lines 15–24, the new value of the solution for the node n_e is:

$$v'_e = \min\{v_e, v_0 + l_e f(n_0)\},$$

where v_0 and v_e are the values of the solution corresponding to n_0 and n_e nodes, and l_e is the length of edge e. Note that if the value of the solution is changed for the adjacent nodes, the position of those nodes in the min-heap has to be updated, Line 22. This process is iterated until there are no nodes in the min-heap.

In our applications we consider a domain bounded by an inner object that defines the inner boundary of the domain, $\partial\Omega_{\text{in}}$, and a smooth outer boundary denoted by $\partial\Omega_{\text{out}}$. In order to find a distance field that takes into account the

Algorithm 1 Solver for the Eikonal equation

```
1: function solveEikonal(Mesh mesh, NodeSet initialNodes)
2:     NodeMinHeap 𝒩 ← ∅
3:     for all Node n ∈ mesh do
4:         if belongs(n,initialNodes) then
5:             setValue(n,0.0)
6:         else
7:             setValue(n,∞)
8:         end if
9:         insert(n,𝒩)
10:    end for
11:    while heapSize(𝒩) > 0 do
12:        Node n_0 ← firstNode(𝒩)
13:        removeNode(n_0,𝒩)
14:        Real v_0 ← getValue(n_0)
15:        for all Edge e adjacent to n_0 do
16:            Real l_e ← length(e)
17:            Node n_e ← oppositeNode(e,n_0)
18:            Real v_e ← getValue(n_e)
19:            Real v'_e ← v_0 + l_e f(n_0)        ▷ Note that in our
    implementation f ≡ 1
20:            if v'_e < v_e then
21:                setValue(n_e,v'_e)
22:                updateHeap(n_e,𝒩)        ▷ Since the value of the
    node has changed
23:            end if
24:        end for
25:    end while
26: end function
```

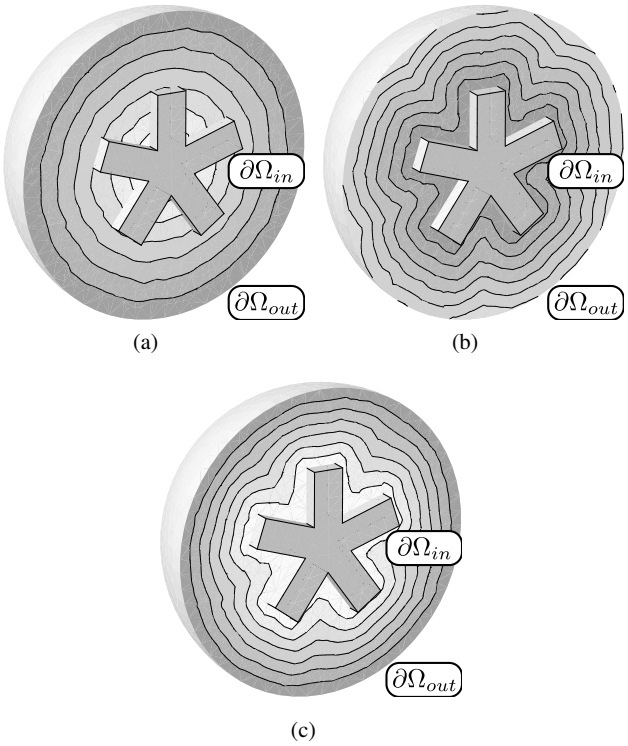


Fig. 6 Distance fields computed in the exterior domain of a five-pointed star: (a) starting from the outer boundary ($\partial\Omega_{\text{out}}$); (b) starting from the inner boundary ($\partial\Omega_{\text{in}}$); and (c) combined distance field.

distance from each inner point to both boundaries we first consider the following problem:

$$\begin{cases} \|\nabla d_{\text{out}}\| = 1 \text{ in } \Omega \\ d_{\text{out}}|_{\partial\Omega_{\text{out}}} = 0, \end{cases} \tag{2}$$

where $d_{\text{out}} \geqslant 0$. The solution of Equation (2) provides the distance to the outer boundary, see Figure 6(a). Second, we consider the problem:

$$\begin{cases} \|\nabla d_{\text{in}}\| = 1 \text{ in } \Omega \\ d_{\text{in}}|_{\partial\Omega_{\text{in}}} = 0, \end{cases} \tag{3}$$

where $d_{\text{in}} \geqslant 0$. The solution of Equation (3) provides the distance to the inner boundary, see Figure 6(b). Note that we use the same tetrahedral mesh to solve both Equations (2) and (3) using Algorithm 1. The *combined distance field*, u, is defined as:

$$u := \frac{d_{\text{out}}}{d_{\text{out}} + d_{\text{in}}}. \tag{4}$$

Note that the combined distance field verifies $0 \leqslant u \leqslant 1$, and at the boundaries of the domain it also verifies that $u|_{\partial\Omega_{\text{out}}} = 0$ and $u|_{\partial\Omega_{\text{in}}} = 1$. Moreover, the contours of u close $\partial\Omega_{\text{out}}$ are similar to d_{out} whereas the contours of u close $\partial\Omega_{\text{in}}$ are similar to d_{in}. That is, the level sets of the combined distance field reproduce the shapes of the inner and outer boundaries close to them. Figure 6(c) presents the combined distance field for the exterior domain of a five-pointed star.

Algorithm 2 Generate level sets ordered from inside to outside

Ensure: LevelSetContainer $\mathcal{L}$
function `ComputeLevelSets`(Mesh *mesh*, Boundary $\partial\Omega_{\text{in}}$, Boundary $\partial\Omega_{\text{out}}$, Int *nOfLevelSets*)
  ScalarField $d_{\text{in}} \leftarrow$ `solveEikonal`(*mesh*, $\partial\Omega_{\text{in}}$)
  ScalarField $d_{\text{out}} \leftarrow$ `solveEikonal`(*mesh*, $\partial\Omega_{\text{out}}$)
  ScalarField $u \leftarrow$ `combineDistanceFields`(d_{in}, d_{out})
  LevelSetContainer $\mathcal{L} \leftarrow$ `getLevelSets`(u, *nOfLevelSets*)
end function

Algorithm 3 Generate hexahedra between level sets

Ensure: Mesh *theMesh*
function `MeshLevels`(LevelSetContainer $\mathcal{L}$)
  Mesh *theMesh* $\leftarrow \emptyset$
  Quad-Mesh $\mathcal{Q}_\ell \leftarrow$ `getQuadMesh`(ℓ_0)
  Features $\mathcal{F}_\ell \leftarrow$ `detectGeometricFeatures`($\mathcal{Q}_\ell$)
  for all LevelSet ℓ in $\mathcal{L}$ **do**
    LevelSet $\ell_{\text{next}} \leftarrow$ `getNextLevelSet`(ℓ)
    `meshFront`(*theMesh*, $\mathcal{Q}_\ell$, $\mathcal{F}_\ell$, ℓ_{next})
    `smoothFront`(*theMesh*, ℓ_{next})
    `refineFront`(*theMesh*, ℓ_{next})
    Quad-Mesh $\mathcal{Q}_\ell \leftarrow$ `getQuadMesh`(ℓ_{next})
    Features $\mathcal{F}_\ell \leftarrow$ `classifyNewFeatures`($\mathcal{Q}_\ell$)
  end for
end function

Finally, we extract m level sets of the combined distance field u. These level sets determine the fronts used to advance the mesh from the meshed inner boundary towards the unmeshed outer boundary. In our implementation, the level sets are represented by means of triangular meshes. Algorithm 2 details the proposed procedure to generate the level sets of the distance field.

4.2 Expanding the Fronts

The expanding process of a quadrilateral mesh defined on a given level to the next one is performed in five steps. First, we classify the entities of the quadrilateral mesh. Second, we use a set of templates to generate an hexahedral mesh between the two levels according to the previous classification. Third, we smooth the hexahedral mesh to improve its quality. Fourth, if needed, we apply a local refinement process to preserve the prescribed element size. Fifth, we classify the features of next level. Algorithm 3 details the proposed procedure to expand the quadrilateral mesh on the inner body to the outer boundary. The initial quadrilateral mesh on the surface of the inner body is generated with the method proposed in [45,46].

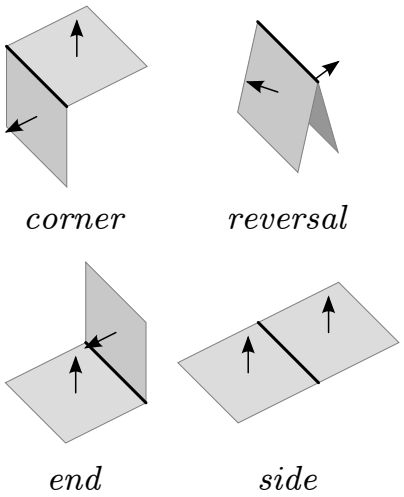


Fig. 7 Edge classification (the normal vectors point towards the advancing direction).

4.2.1 Feature Vertices and Feature Edges

Given an initial quadrilateral mesh on the surface of the inner boundary, we have to detect its geometric features. These geometric features define the topology of the mesh for the current level. First, the edges of the quad-mesh are classified according to the angle, ϕ, defined by the outer normal of its adjacent faces as (see Figure 7):

$$\begin{cases} \textit{corner} \text{ edge} & \pi/4 \leq \phi < 3\pi/4, \\ \textit{reversal} \text{ edge} & 3\pi/4 \leq \phi < 5\pi/4, \\ \textit{end} \text{ edge} & 5\pi/4 \leq \phi < 7\pi/4, \\ \textit{side} \text{ edge} & \text{otherwise.} \end{cases} \tag{5}$$

We define a *local feature edge* as an edge that is not classified as *side*. That is, an edge that locally defines a geometric feature of the quadrilateral mesh.

The nodes of the quadrilateral mesh are classified according to the number of adjacent local feature edges.

$$\begin{cases} \text{2D node} & \text{0 adjacent local feature edges,} \\ \text{1D node} & \text{1 or 2 adjacent local feature edges,} \\ \text{0D node} & \text{3 or more adjacent local feature edges.} \end{cases}$$

The main idea is that 0D nodes represent a vertex, 1D nodes belong to a curve and 2D nodes belong to a surface. Figure 8 presents a quadrilateral mesh with its feature edges marked with thick line. In addition, 0D nodes are depicted with gray circles and 1D nodes using white circles. All other nodes are classified as 2D nodes and are marked with black circles. For the special case where a node is adjacent to exactly one local feature edge, node is classified as *semi-vertex* and the adjacent local feature edge as a local *semi-edge*, see Figure 9.

The above classification of nodes and edges locally determines the topology of the hexahedral mesh. That is, we have obtained a local model of the features contained in the mesh. However, to advance the fronts taking into account the global organization of the features we have to generate a global model. For this purpose, we define a *feature vertex* as a node classified as 0D node. In addition, we define a *global feature edge* as a continuous path of local feature edges that connects two feature vertices. To illustrate these definitions, Figure 10 shows a quadrilateral mesh and its corresponding feature vertices and global feature edges. While Algorithm 4 presents the proposed procedure to compute the global feature edges of a quadrilateral mesh given the local feature edges and the feature vertices. The idea of the algorithm is

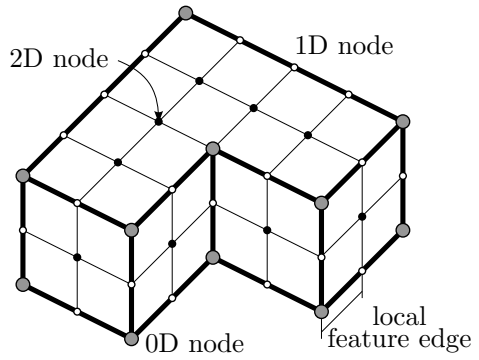


Fig. 8 Detected local feature edges (thick line) and its corresponding node classification: 0D nodes in gray circles; 1D nodes in white circles; 2D nodes in black circles.

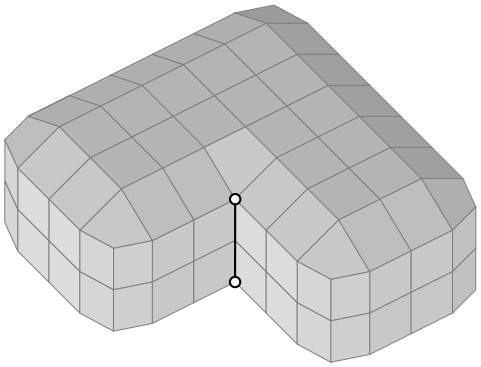


Fig. 9 A semi-edge (thick line) and corresponding semi-vertices (white circles) for a quadrilateral mesh.

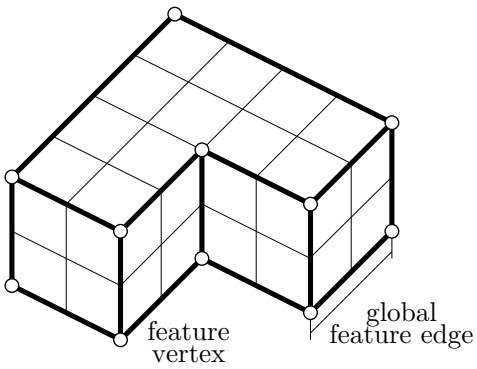


Fig. 10 Feature edges (thick line) and feature vertices (white circles) computed from the local feature edges.

Algorithm 4 Compute global feature edges

```
Ensure: GlobalFeatureEdgeList ℰ
 1: function computeGlobalFeatureEdges(FeatureVertexList
    𝒱)
 2:     GlobalFeatureEdgeList ℰ ← ∅
 3:     for all FeatureVertex ν ∈ 𝒱 do
 4:         for all LocalFeatureEdge e adjacent to ν do
 5:             if not ispreviouslyUsed(e) then
 6:                 FeatureEdge ε ← ∅
 7:                 Node n ← getNode(ν)
 8:                 repeat
 9:                     insertEdge(e,ε)
10:                     Node n ← oppositeNode(e,n)
11:                     e ← getNextLocalFeatureEdge(n, e)
12:                 until isFeatureVertex(n)
13:                 insert(ε,ℰ)
14:             end if
15:         end for
16:     end for
17: end function
```

to loop on the feature vertices (Lines 3–15) and, for each feature vertex, compute its adjacent global feature edges (5–14). Once all the feature vertices are traversed, the global feature edges are generated. In addition, if a local feature edge is a semi-edge, the global feature edge is also classified as semi-edge. The main idea is that the local feature model defines the topology of the mesh for the current level while the global model will maintain the coherence of the local model between two consecutive levels.

4.2.2 Meshing the Fronts

From a quadrilateral surface mesh of the inner boundary we want to generate an unstructured hexahedral mesh of the domain without prescribing a quadrilateral surface mesh on the outer boundary. Each level-set computed in the initial tetrahedral mesh of the volume will delimit a new front of hexahedral elements. Therefore, we have to describe the procedure to expand a quadrilateral mesh on level set ℓ to level set $\ell + 1$. This process is performed in three stages:

(i) *Generation of new hexahedra adjacent to feature vertices*. The advancing template at each feature vertex is determined by the number of adjacent local feature edges classified as *end*, *reversal* and *corner*. The adjacent edges determine the topology of the mesh at each particular feature vertex. Figure 11 illustrates the templates used to expand the feature nodes of the quadrilateral mesh. Note that there exist additional templates that address other arrangements of feature edges. However, they are not considered in this work.

(ii) *Generation of new hexahedra adjacent to local feature edges*. The classification of each local feature edge determines the used template to expand the layer to the next level. Figure 12 presents the templates used to expand feature edges. Note that local feature edges that were used in the previous step do not have to be expanded.

(iii) *Generation of new hexahedra adjacent to quadrilateral faces*. For each face of the quadrilateral mesh that has not been expanded, apply the extrusion template presented in Figure 13. Note that some of the quadrilateral faces were used in the previous steps and, for this reason, they do not have to be expanded.

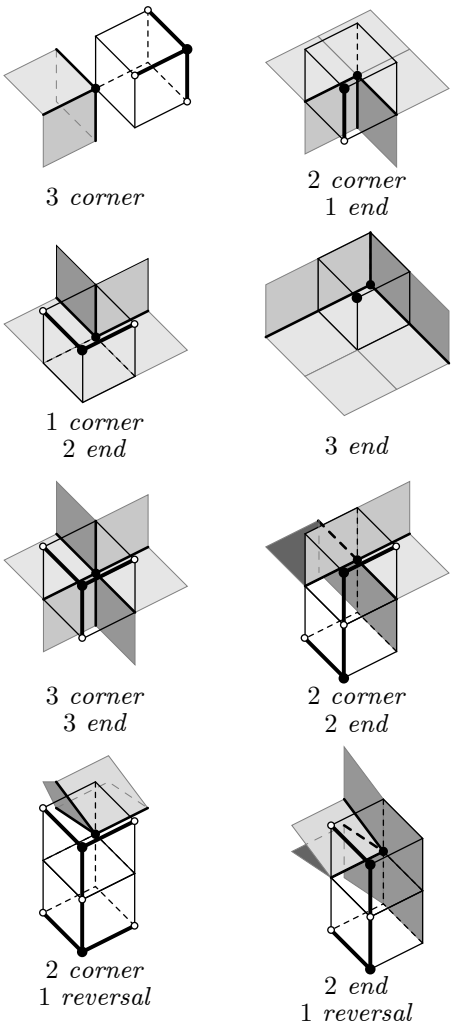


Fig. 11 Advancing vertex templates. Previous level mesh in shaded gray, feature vertices in black circles, and local feature edges in thick lines. Next level feature edges in thick black line and next level 0D nodes in black circles and 1D nodes in white circles.

Note that *end* and *corner* edge templates are advancing templates that allow to increase (corner template) or reduce (end template) the number of hexahedra from one level to the next one. However, the templates introduced in Figures 11 and 12 do not deal with semi-edges and semi-vertices. These templates cannot be applied around semi-edges and semi-vertices, because they will lead to a non-conformal mesh. For this reason, we introduce five new templates for the semi-edge and semi-vertex features that allow to increase

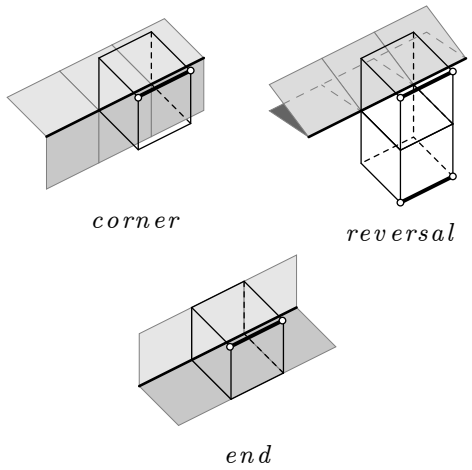


Fig. 12 Advancing edge templates. Previous level mesh in shaded gray, feature vertices in black circles, and local feature edges in thick lines. Next level feature edges in thick black line and 1D nodes in white circles.

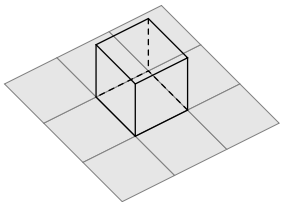
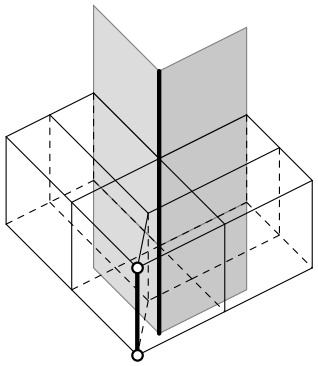


Fig. 13 Advancing face template. Previous level mesh in shaded gray.

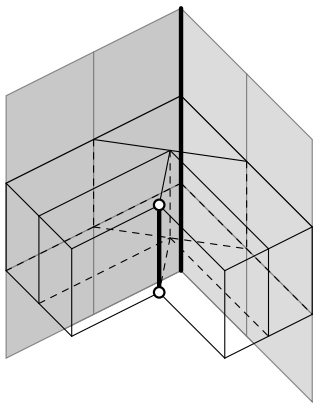
the element quality around these geometric features. Three of them are used to expand semi-edges and two of them to advance semi-vertices. Figure 14 presents the advancing templates corresponding to semi-edges, where the previous quadrilaterals are shaded and the new hexahedra are not shaded. Specifically, Figures 14(a) and 14(b) show the advancing templates corresponding to a semi-edge classified as *corner* and a semi-edge classified as *end*, respectively. These advancing templates introduce an additional layer of elements around the semi-edges. For this reason, the template presented in Figure 14(c) is introduced in order to provide the transition between the edges with one level and the edges with two levels. This template can be thought as the advancing template defined for a special semi-edge. This semi-edge is called *side* semi-edge.

The advancing templates for semi-vertices are presented in Figure 15, where previous quadrilaterals are shaded and new hexahedra are not shaded. Specifically, Figure 15(a) presents the advancing template for a semi-vertex adjacent to a semi-edge classified as *corner*, while Figure 15(b) shows the advancing template for a semi-vertex adjacent to a semi-edge classified as *end*. These templates allow to generate a hexahedral mesh around semi-edges.

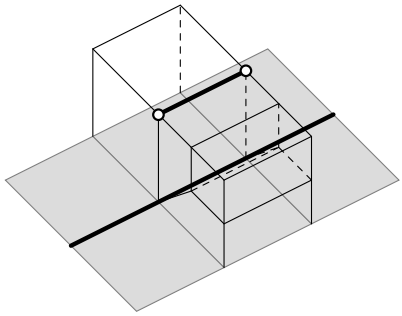
In order to generate the hexahedral mesh around these features, we first have to detect the *side* semi-edges that determine the transition from the area with two sub-levels and



(a)



(b)



(c)

Fig. 14 Advancing templates for semi-edges: (a) advancing template for a semi-edge classified as *corner*; (b) advancing template for a semi-edge classified as *end*; and (c) advancing template for a semi-edge classified as *side*.

the area with one level, see Figure 9. Two side semi-edges are propagated from each of the semi-vertices, see Figure 16. The new semi-edges are propagated until they *hit* a feature edge. At this point, the appropriate templates are used in order to generate the hexahedral mesh for the current level.

As the advancing templates are applied, the quadrilateral mesh of the next level is generated. Hence, each advancing

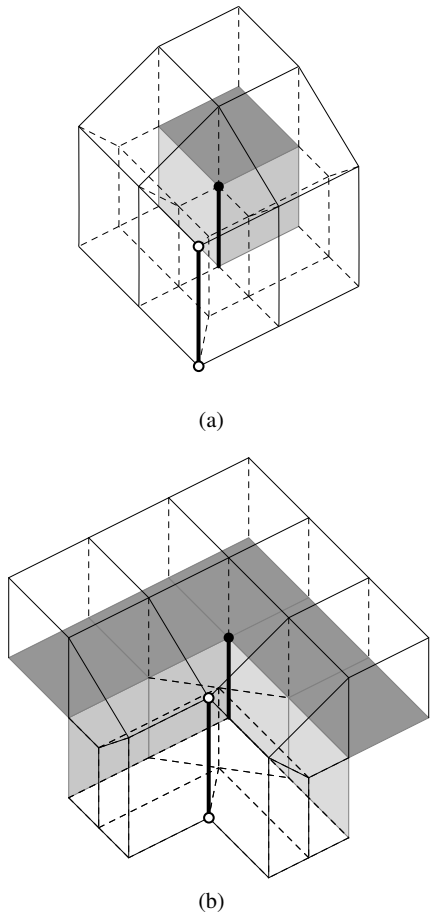


Fig. 15 Advancing templates for semi-vertices: (a) semi-vertex adjacent to a *corner* semi-edge; and (b) semi-vertex adjacent to an *end* semi-edge.

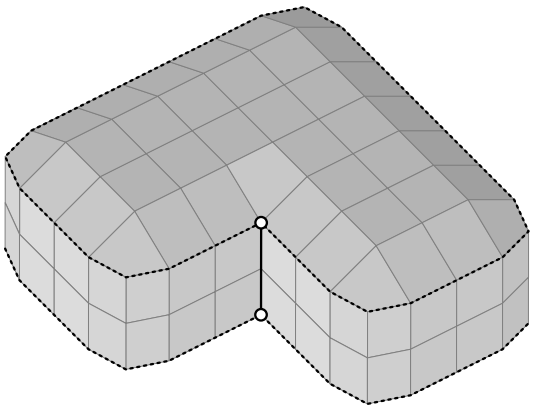


Fig. 16 Process of *side* semi-edges creation (dotted line) from an initial *end* semi-edge (thick line) and its corresponding semi-vertices (white circles).

front template defines the local feature edges and the feature vertices of the next level. For this reason, at the same time as the advancing front process is performed, it is possible to classify the next level nodes and to identify the local feature edges of the next level. Recall that if the nodes of a level are classified, we have also identified the feature vertices of that level since 0D nodes correspond to feature vertices. Figures 11 to 15 present the classification of the nodes of the next level for each type of advancing template. In these figures, black circles represent 0D nodes of the next level, white circles represent 1D nodes of the next level and the remaining nodes are 2D nodes. In addition Figures 11 to 15 show the local feature edges of the next level in thick lines. Further analysis is needed to classify them as *corner*, *reversal*, *end* or *side*, see Section 4.2.4.

4.2.3 Smoothing the Fronts

When all the hexahedra of the current level are generated, the position of the newly created nodes may not define high-quality elements. For this reason, a smoothing technique is applied to the current front of hexahedra. For each node, the proposed smoothing process is performed in two steps.

1. *Re-location of the node*. The node is located to the position determined by the untangling and smoothing procedure detailed in [47,48]. The nodes of the previous levels are fixed.
2. *Projection of the node to the current level set surface*. The position computed in the first step is not located on the surface defined by the level set. For this reason, the node has to be projected to the level set. To this end, we find the triangle of the level set that is the closest to the node and project the node to that triangle. Note that mid-level nodes introduced by semi-edge and semi-vertex templates do not have to be projected onto next level set.

Experience has shown that four or five iterations of this process are sufficient to improve the quality of the mesh. In addition, this process also increases the robustness of the receding front method.

4.2.4 Refining the Fronts

Once the position of new nodes is improved, the size of the new hexahedra may differ from the prescribed element size. In particular, if the size of the new hexahedra is bigger than the prescribed element size, a local refinement process is performed at each level. This process is based on the edges of the mesh. The main advantage of this algorithm is that it provides anisotropic refinement on the hexahedral mesh. That is, the mesh is only refined on the required directions. In order to select the edges to be refined, let h and l_e be the prescribed element size and the length of edge e, respectively. This edge is marked to be refined if:

$$\left| \frac{l_e}{3} - h \right| \leq |l_e - h| \,. \tag{6}$$

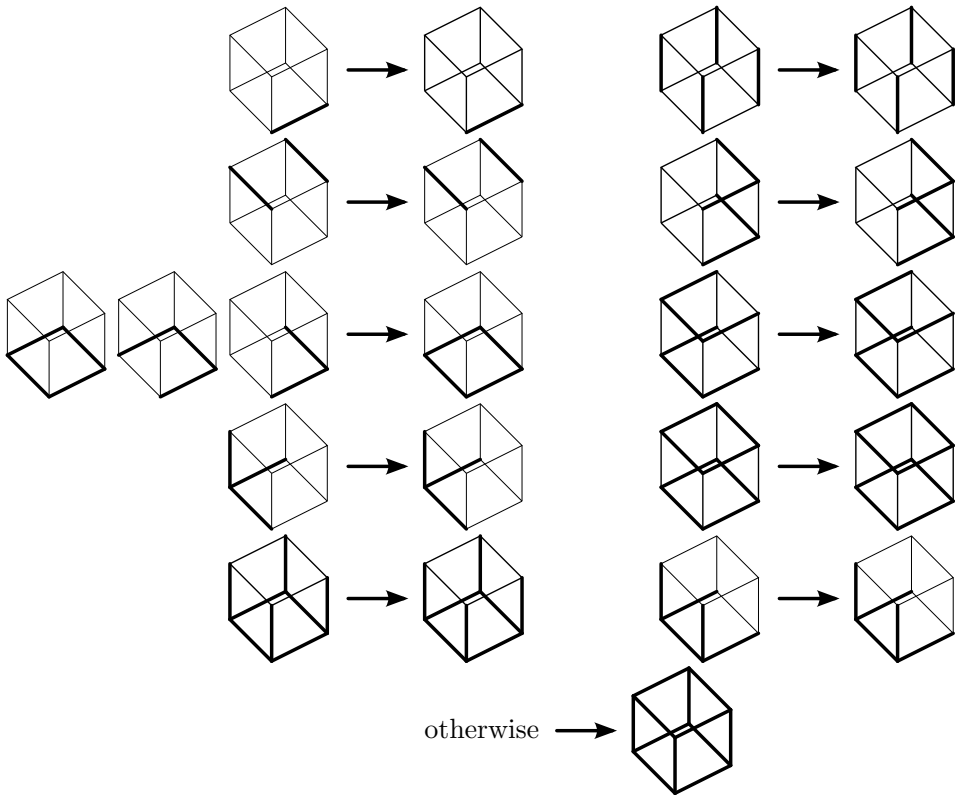


Fig. 17 Classification of the templates used during the refining process according to the marked edges.

That is, an edge is refined if the refined edge length better approximates the prescribed element size than the current edge length. Since the refining process is based on templates that divide the selected edges in three parts (sub-edges), the refined element size is $l_e/3$. When the edges to be refined are selected, adjacent hexahedra are replaced by a given template of hexahedra depending on the marked edges. To this end, we use the method proposed in [49] where the templates introduced in [9] and [10] are adapted and extended to marked edges. Given a set of marked edges to be refined, the algorithm substitutes each hexahedron by a template of hexahedra depending on the marked edges. The edge refinement algorithm proceeds as follows:

1. *Template expansion*. Some combinations of marked edges cannot be substituted by a given template. For this reason, additional edges have to be marked in order to apply a valid template. For each hexahedron that contains marked edges, we check if an appropriate template exists. If no template exists, we mark the additional edges as presented in Figure 17. This process is iterated until all hexahedra can be substituted by an existing template. According to our experience, the number of additional marked hexahedra is less than the 10% of the number of initial marked hexahedra.
2. *Template substitution*. In this step, each hexahedron is refined using a template of new hexahedra. Figure 18 shows the corresponding template substitution for each combination of marked edges.

Note that when templates are applied in order to replace an hexahedron, local feature edges and feature vertices have to be replaced by the refined ones.

4.2.5 Classifying the new features

When the elements for the current level are refined, we still have to compute the global feature edges of the next level. To identify the global feature edges, we first apply Algorithm 4 to the quadrilateral mesh generated on the next level. We can apply it because the local feature edges have been already identified. Recall that in Section 4.2.2 we have detailed how to identify (not classify) the local feature edges from the meshing templates. Second, we classify the global feature edges. To this end, we compute the average angle of the adjacent faces of its local feature edges, $\bar{\phi}$. Then, according to the angle $\bar{\phi}$, global feature edges are classified as:

$$\begin{cases} \textit{corner} \text{ edge} & \pi/4 \leq \bar{\phi} < 3\pi/4, \\ \textit{reversal} \text{ edge} & 3\pi/4 \leq \bar{\phi} < 5\pi/4, \\ \textit{end} \text{ edge} & 5\pi/4 \leq \bar{\phi} < 1.85\pi, \\ \textit{side} \text{ edge} & \text{otherwise.} \end{cases} \tag{7}$$

Note that the only value that has been modified from (5) corresponds to *end* global feature edges. In this way, we avoid

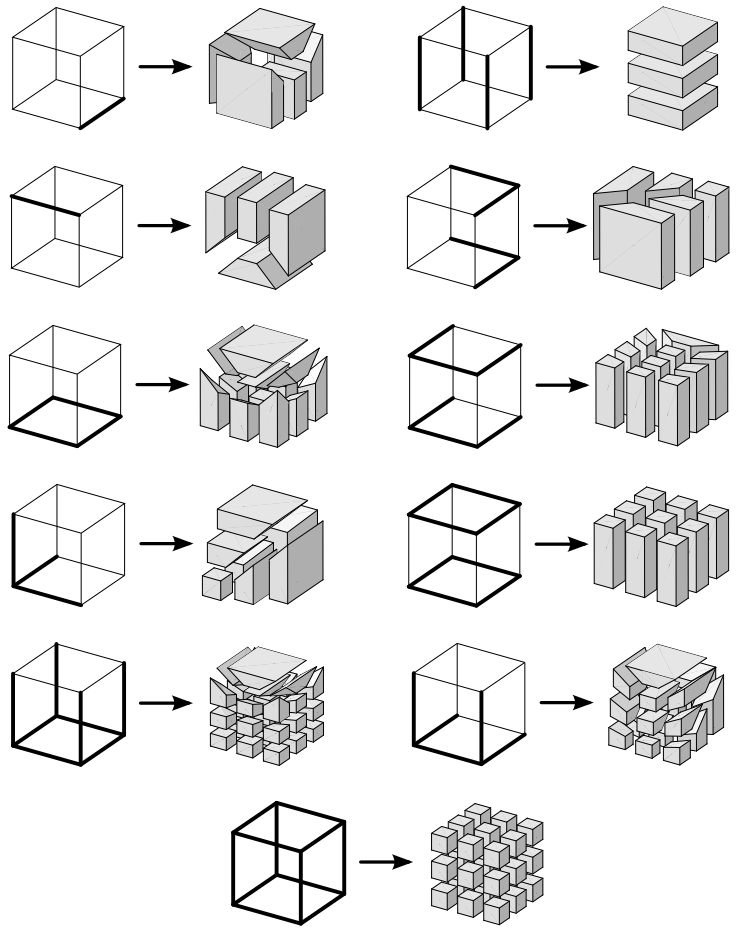


Fig. 18 Substitution templates for each combination of marked edges.

that *end* global feature edges are classified as *side* too early in the algorithm. Then, the local feature edges are classified as the containing global feature edge.

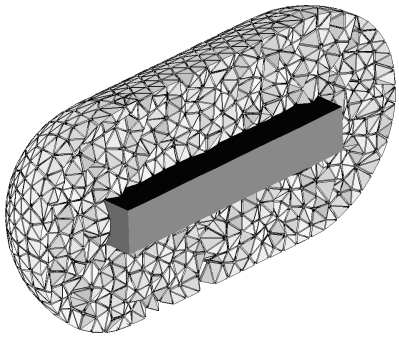
The algorithm of front meshing is iterated until all the fronts are discretized. That is, until the outer boundary is reached.

5 Examples

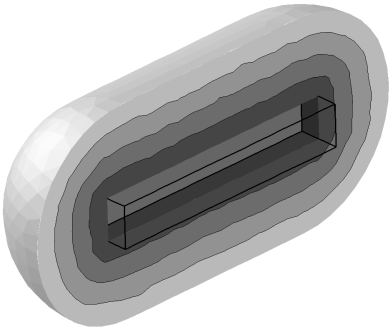
In this section we present four examples where the receding front method is applied to the discretization of the exterior domain of four simple geometries. In all the cases the starting mesh is a quadrilateral mesh of the inner surface. The user input is the element size of the quadrilateral mesh and the number of levels of the mesh. In these examples, we illustrate the process of the receding front method with geometries that contain different types of feature edges. In addition, we show the need of a refining process to avoid bigger elements that the prescribed element size. Finally, we sketch a procedure to generate stretched elements along the advancing direction.

Long box. The goal of the first example is to illustrate the steps of the method for a simple geometry. To this end, we present the mesh for the exterior domain of a long box located inside a smooth domain, see Figure 19. Note that the inner boundary only contains global feature edges classified as *corner*, see Section 4.1. Figure 19(a) presents the tetrahedral mesh used to compute the solution of both Eikonal equations. Figure 19(b) presents four pre-computed fronts as detailed in Section 4.1. Figure 19(c) shows a general view of the hexahedral mesh, while Figure 19(d) illustrates a longitudinal cut of the mesh. Although the quadrilateral surface mesh of the inner box is structured, the final mesh contains unstructured nodes both in the interior and on the boundary of the mesh. For instance, in Figure 19(c) we highlight a node with three adjacent hexahedra that comes from a vertex of the long box. In addition, Figure 19(d) shows an inner node with six adjacent hexahedra that appears when the global feature edge is classified from *corner* to *side*.

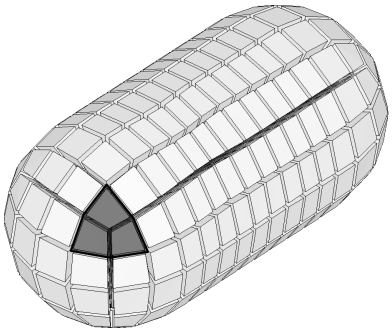
Five-pointed star. The second example presents the mesh generated for a domain delimited by a star placed inside a sphere. In this case the definition of the domain contains feature edges classified as *corner* and *end*. The final



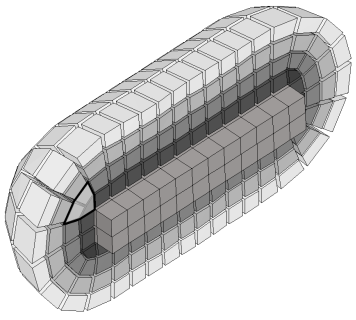
(a)



(b)

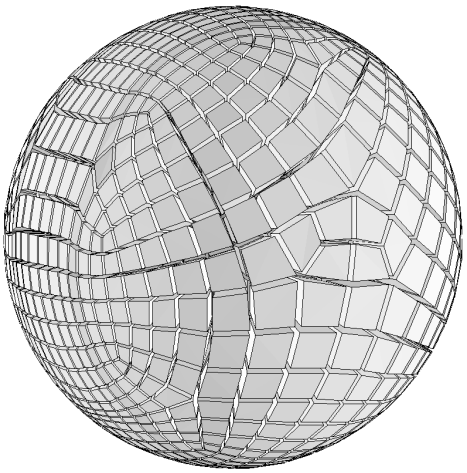


(c)

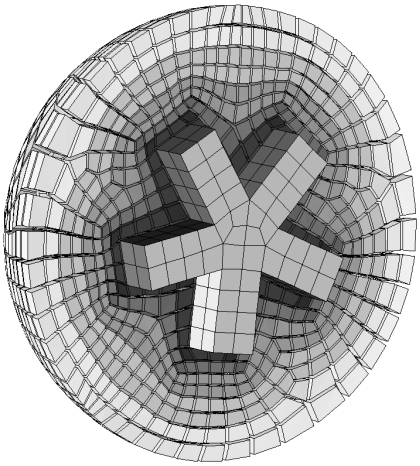


(d)

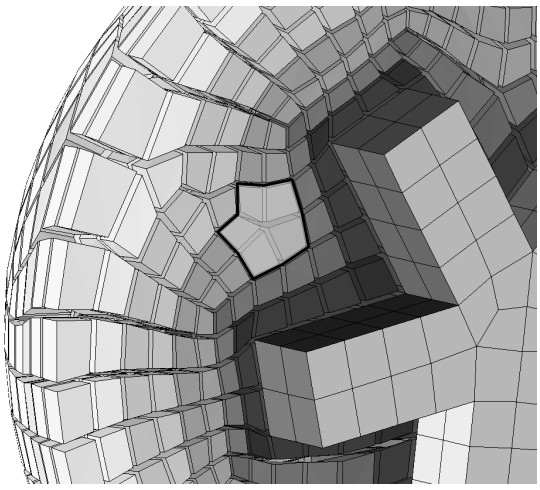
Fig. 19 Hexahedral mesh for the exterior domain of the long box: (a) tetrahedral mesh used to solve the Eikonal equation; (b) level sets of the combined distance field; (c) general view of the hexahedral mesh; and (d) longitudinal cut of the hexahedral mesh.



(a)

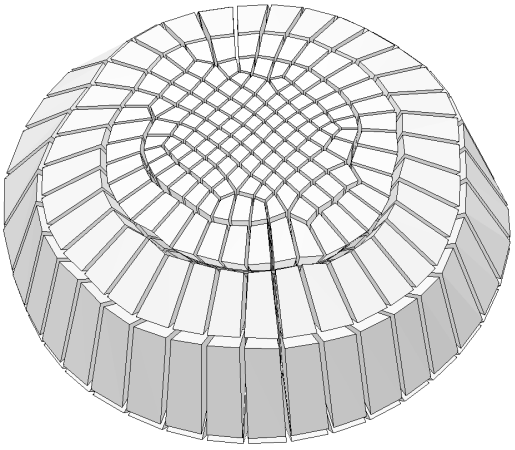


(b)

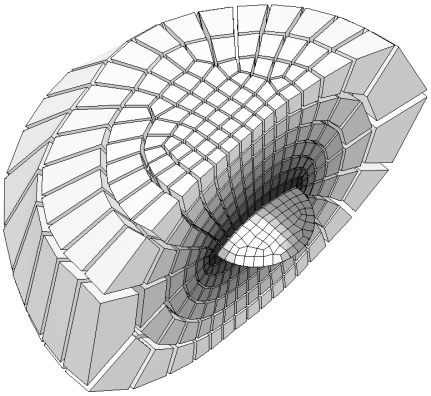


(c)

Fig. 20 Hexahedral mesh for the exterior domain of the pentagonal star: (a) general view; (b) vertical cut; and (c) detail of an *end* semi-edge region.



(a)

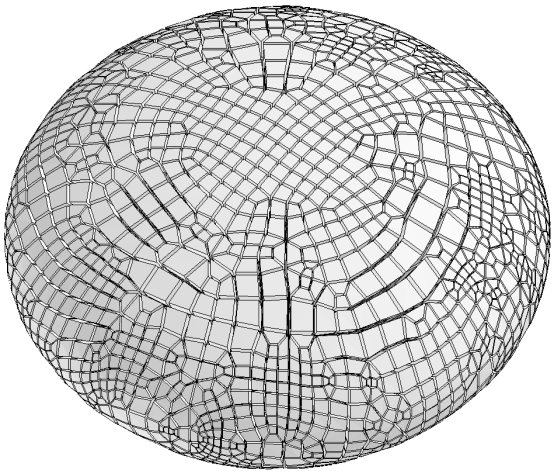


(b)

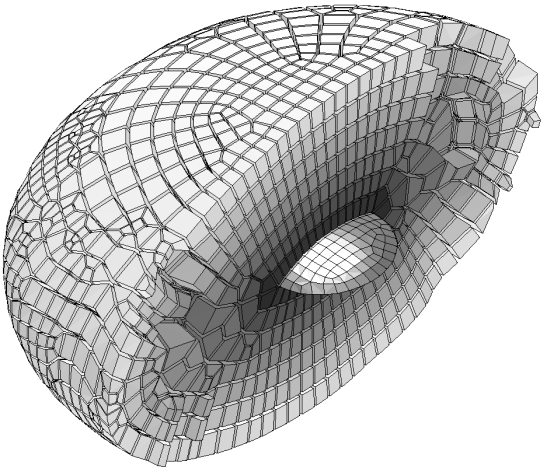
Fig. 21 Hexahedral mesh without local refinement for the exterior domain of the flat object with a reversal feature: (a) general view of the outer boundary mesh; and (b) longitudinal cut.

mesh is composed by five fronts (Figure 20(b)): two of them with a single layer of hexahedra, and three of them a double layer of hexahedra (due to semi-edge advancing templates). Figure 20(c) presents a detail of a region where an *end* semi-edge template is used since the only global feature edges remaining are those classified as *end*. Note that the expansion of the seed surface mesh generates unstructured elements along the advancing direction. The change of topology allows to generate high-quality elements that follow the pre-computed fronts.

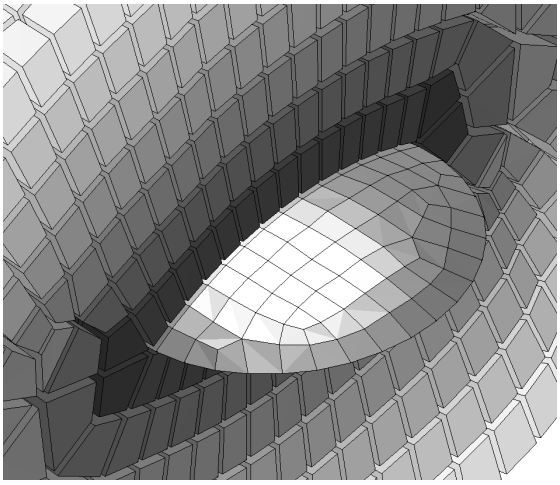
Flat object. The objective of the third example is to show that using a refinement procedure we can respect the prescribed element size in the final mesh. To this end, we discretize a domain delimited by a flat object inside an ellipsoid, which only contains feature edges classified as *reversal*. First, we generate a hexahedral mesh without using the local refinement process described in Section 4.2.4, see Figure 21. Note that the obtained element size near the outer



(a)



(b)



(c)

Fig. 22 Hexahedral mesh with local refinement for the exterior domain of the flat object with a reversal feature: (a) general view of the outer boundary mesh; (b) longitudinal cut; and (c) detail of the inner levels.

boundary is greater than the obtained element size near the inner boundary. In order to preserve the prescribed element size, in each level we perform a local refinement, see Figure 22. The result shows that the final mesh reproduces with more fidelity the prescribed element size. Note that in both cases an unstructured mesh is obtained.

Space capsule. One of the advantages of the proposed approach is that it is straightforward to stretch the elements in the normal direction of the fronts. To this end, we use a blending function [50] that modifies the combined distance field u introduced in Equation (4):

$$\begin{cases} \widetilde{u} = \dfrac{d_2}{d_1} u & 0 \leq u \leq d_1 \\ \widetilde{u} = d_2 + (1 - d_2) \dfrac{2^{\alpha(u-d_1)/(1-d_1)} - 1}{2^{\alpha} - 1} & d_1 \leq u \leq 1 \end{cases} \tag{8}$$

where $0 \leq d_1 \leq 1$, $0 \leq d_2 \leq 1$ and α is a real parameter. In order to illustrate the behavior of the blending function presented in (8), Figure 24 shows the new level sets distribution using $d_1 = 0.5$, $d_2 = 0.7$ and $\alpha = 5$.

The objective of the fourth example is to generate a stretched mesh for the exterior domain of a space capsule. To this end, we generate a boundary layer by using the blending function (8). The final mesh, presented in Figure 24, contains 28 levels. Figure 24(a) shows a general view of the outer boundary of the final mesh. Figure 24(b) presents a cut of the mesh and Figure 24(c) shows a detail of the boundary layer that follows the modified distance field.

Aircraft. This example presents an unstructured hexahedral mesh for the exterior domain of an aircraft. This mesh is automatically generated using the receding front method. The final mesh is composed by twenty levels. Figures 25(a) and 25(b) show a cross section of the final mesh. Note that the final mesh correctly adapts to the shape of the inner body and is composed by high-quality elements.

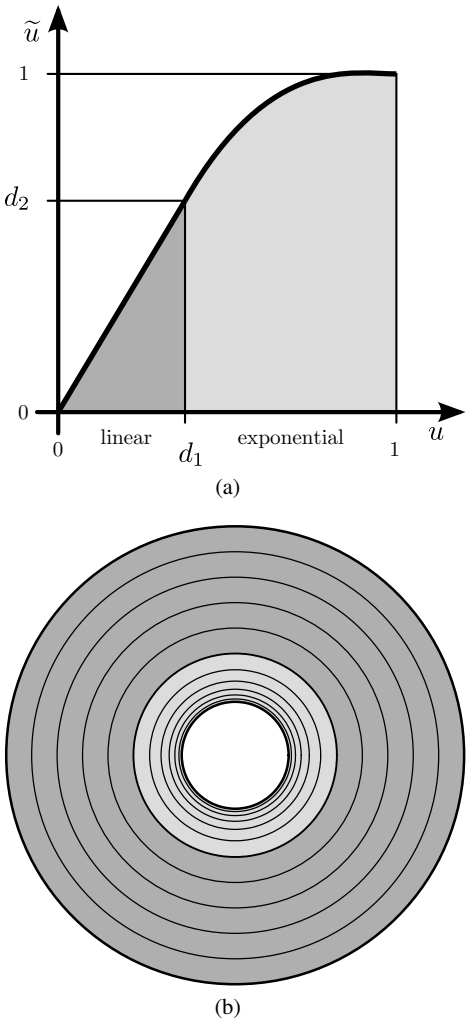
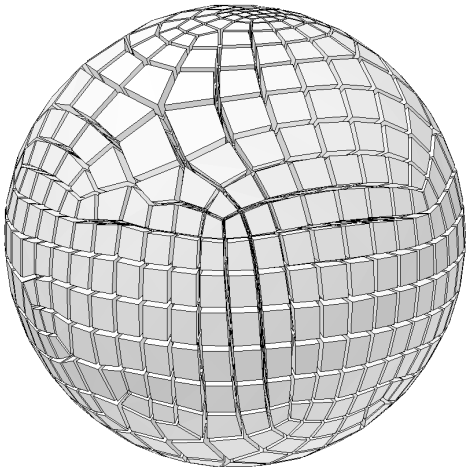


Fig. 23 Blending function composed by a linear part (dark gray) and an exponential part (light gray).

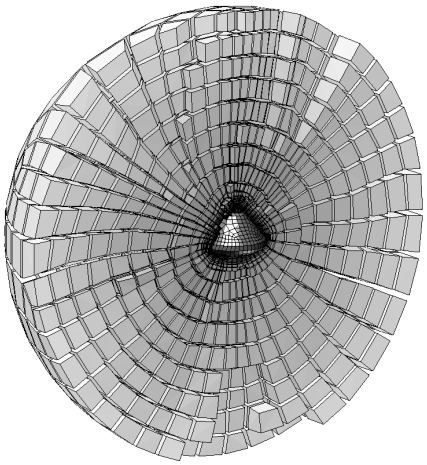
6 Summary and Future Work

In this work we have proposed the receding front method, a new approach for generating unstructured hexahedral meshes of exterior domains. Specifically, the two main contributions of this work are to pre-compute the meshing fronts by combining two solutions of the Eikonal equation, and to advance unstructured hexahedral elements from inside to outside (recede) guided by the pre-computed fronts. The former allows us to obtain meshes that reproduce the shape of the domain close to the outer boundary. The latter allows us to avoid the collision of constrained meshing fronts. The preliminary results presented in this work show the possibilities of the receding front method applied to the unstructured hexahedral mesh generation for exterior domains where *corner*, *reversal* and *end* feature edges and semi-edges are present. Moreover, we show that it is straightforward to obtain stretched meshes along the normal direction of the domain boundaries. We have implemented the proposed method in the ez4u meshing environment [6,51,52].

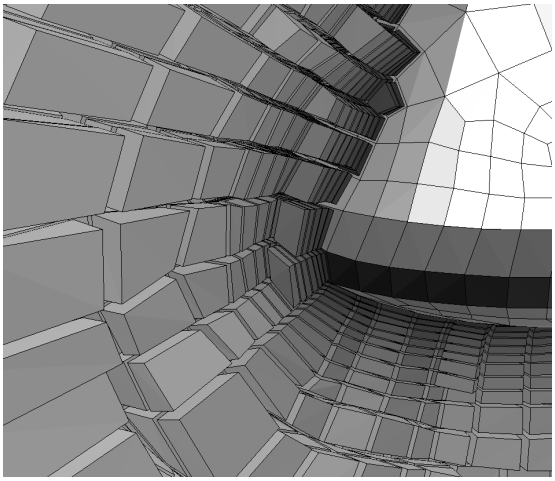
Our long-term goal is to obtain a general-purpose unstructured hexahedral mesh generator based on the receding front method. In this sense, the first implementation of the method presents several issues that should be investigated and solved in the near future. First, we are currently including additional advancing and refinement templates. These templates allow us to improve the quality of the meshes obtained by advancing the elements from one layer to the following one. Second, we are developing a local coarsening process to preserve the prescribed element size when the elements of the mesh are too small. Third, we want to extend the presented approach to mesh the exterior domain of



(a)

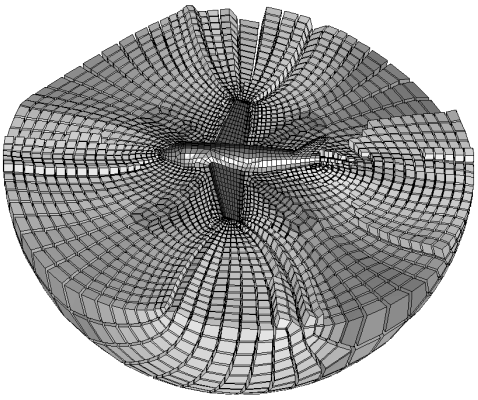


(b)

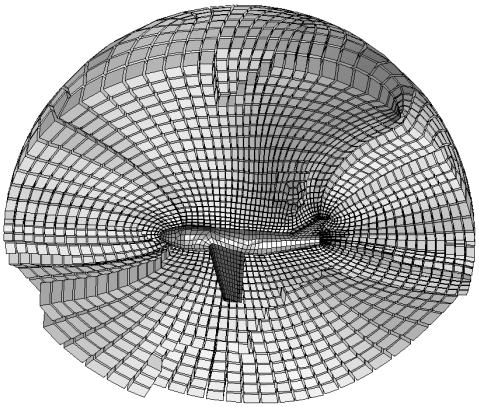


(c)

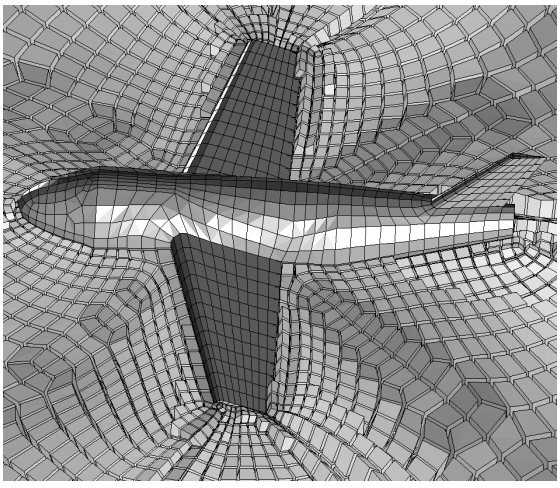
Fig. 24 Hexahedral mesh for the exterior domain of the space capsule: (a) general view of the outer boundary mesh; (b) longitudinal cut; and (c) detail of the inner levels.



(a)



(b)



(c)

Fig. 25 Hexahedral mesh for the exterior domain of an aircraft: (a) view of the interior of the mesh; (b) alternative view of the interior of the mesh; and (c) detail of an unstructured area near the aircraft.

several objects and objects with holes, for instance a torus inside a sphere. Fourth, we want to apply the exterior domain meshing tool to outer boundaries with feature curves and vertices. To this end, we need to develop an imprinting technique that allows propagating through the fronts the features of the outer boundary towards the inner boundary. These imprints would determine a decomposition of the domain in sub-volumes that connect the outer boundary with the inner boundary. Then, we can restrict the receding front method to each one of the sub-volumes to advance layer-by-layer unstructured hexahedra from the inner mesh to the outer boundary. The resulting hex-meshing primitive would respect the boundary features and would be equivalent to a fully unstructured sweeping (regular sweeping is semi-structured). Fifth, we will analyze how to deal with narrow regions where the thickness of the part is significantly smaller than the surrounding volume. Since our approach generates the same number of levels in the whole domain, the distance between two consecutive level sets is variable. Therefore, it could be interesting to generate different number of hexahedral layers in different regions bounded by two consecutive level sets. To this end, we will investigate how to discontinue a layer and connect it to the boundary in one part of the model, but continue advancing the fronts in other parts. Sixth, we have to investigate how to automatically generate an inner hexahedral mesh that approximately reproduces the skeleton of the domain. To this end, we have considered to use a similar technique to the one proposed in [25,26]. Then, we can obtain an automatic unstructured hexahedral mesh generator by means of advancing the fronts from inside-to-outside with the receding front method. Finally, we have to analyze how the accuracy of the Eikonal equation solution influences in the resulting hexahedral mesh.

References

1. S.J. Owen. A survey for unstructured mesh generation technology. In *7th International Meshing Roundtable*, pages 239–267, 1998.
2. T.D. Blacker. Automated conformal hexahedral meshing constraints, challenges and opportunities. *Engineering with Computers*, 17(3):201–210, 2001.
3. T.J. Tautges. The generation of hexahedral meshes for assembly geometry: survey and progress. *International Journal for Numerical Methods in Engineering*, 50(12):2617–2642, 2001.
4. T.J. Baker. Mesh generation: Art or science? *Progress in Aerospace Sciences*, 41(1):29–63, 2005.
5. J.F. Shepherd. *Topologic and geometric constraint-based hexahedral mesh generation*. PhD thesis, The University of Utah, 2007.
6. X. Roca. *Paving the path towards automatic hexahedral mesh generation*. PhD thesis, Universitat Politècnica de Catalunya, 2009.
7. R. Schneiders and R. Bünten. Automatic generation of hexahedral finite element meshes. *Computer Aided Geometric Design*, 12(7):693–707, 1995.
8. R. Schneiders. A grid-based algorithm for the generation of hexahedral element meshes. *Engineering with Computers*, 12(3):168–177, 1996.
9. Y. Zhang, C. Bajaj, and B.S. Sohn. 3D finite element meshing from imaging data. *Computer Methods in Applied Mechanics and Engineering*, 194(48-49):5083–5106, 2005.
10. Y. Zhang and C. Bajaj. Adaptive and quality quadrilateral/hexahedral meshing from volumetric data. *Computer Methods in Applied Mechanics and Engineering*, 195(9-12):942–960, 2006.
11. J. Qian and Y. Zhang. Sharp Feature Preservation in Octree-Based Hexahedral Mesh Generation for CAD Assembly Models. In *Proceedings of the 19th International Meshing Roundtable*, pages 243–262, 2010.
12. T.D. Blacker and R.J. Meyers. Seams and wedges in Plastering: a 3-D hexahedral mesh generation algorithm. *Engineering with computers*, 9(2):83–93, 1993.
13. M.L. Staten, S.J. Owen, and T.D. Blacker. Unconstrained paving and plastering: A new idea for all hexahedral mesh generation. In *14th International Meshing Roundtable*, 2005.
14. M.L. Staten, R.A. Kerr, S.J. Owen, and T.D. Blacker. Unconstrained paving and plastering: Progress update. In *Proceedings, 15th International Meshing Roundtable*, pages 469–486. Springer, 2006.
15. M.L. Staten, R.A. Kerr, S.J. Owen, T.D. Blacker, M. Stupazzini, and K. Shimada. Unconstrained plastering-hexahedral mesh generation via advancing-front geometry decomposition. *International Journal for Numerical Methods in Engineering*, 81(2):135–171, 2010.
16. S. Meshkat and D. Talmor. Generating a mixed mesh of hexahedra, pentahedra and tetrahedra from an underlying tetrahedral mesh. *International Journal for Numerical Methods in Engineering*, 49(1-2):17–30, 2000.
17. S.J. Owen and S. Saigal. H-Morph: an indirect approach to advancing front hex meshing. *International Journal for Numerical Methods in Engineering*, 49(1-2):289–312, 2000.
18. X. Roca and J. Sarrate. Local dual contributions on simplices: A tool for block meshing. In *Proceedings of the 17th International Meshing Roundtable*, 2008.
19. X. Roca and J. Sarrate. Local dual contributions: Representing dual surfaces for block meshing. *International Journal for Numerical Methods in Engineering*, 83(6):709–740, 2010.
20. N. Kowalski, F. Ledoux, M.L. Staten, and S.J. Owen. Fun sheet matching - automatic generation of block-structured hexahedral mesh using fundamental sheets. In *10th USNCCM*, 2009.
21. X. Roca, E. Ruiz-Gironés, and J. Sarrate. Receding Front Method: a New Approach Applied to Generate Hexahedral Meshes of Outer Domains. In *Proceedings of the 19th International Meshing Roundtable*, 2010.
22. E. Ruiz-Gironés. *Automatic hexahedral meshing algorithms: from structured to unstructured meshes*. PhD thesis, Universitat Politècnica de Catalunya, 2011.
23. J.A. Sethian. Curvature flow and entropy conditions applied to grid generation. *J. Comp. Phys*, 1994.
24. Y. Wang, F. Guibault, and R. Camarero. Eikonal equation-based front propagation for arbitrary complex configurations. *International Journal for Numerical Methods in Engineering*, 73(2):226–247, 2007.
25. H. Xia and P.G. Tucker. Finite volume distance field and its application to medial axis transforms. *International Journal for Numerical Methods in Engineering*, 82(1):114–134, 2009.
26. H. Xia and P.G. Tucker. Distance solutions for medial axis transform. In *Proceedings of the 18th International Meshing Roundtable*, pages 247–265, 2009.
27. T. K. H. Tam and C. G. Armstrong. 2D finite element mesh generation by medical axis subdivision. *Advances in engineering software and workstations*, 13(5-6):313–324, 1991.
28. M.A. Price and C. G. Armstrong. Hexahedral mesh generation by medial surface subdivision: Part II. Solids with flat and concave edges. *International Journal for Numerical Methods in Engineering*, 40(1):111–136, 1997.

29. M.A. Price, C. G. Armstrong, and M.A. Sabin. Hexahedral mesh generation by medial surface subdivision: Part I. Solids with convex edges. *International Journal for Numerical Methods in Engineering*, 38(19):3335–3359, 1995.
30. A. Sheffer, M. Etzion, A. Rappoport, and M. Bercovier. Hexahedral mesh generation using the embedded Voronoi graph. *Engineering with Computers*, 15(3):248–262, 1999.
31. A. Sheffer and M. Bercovier. Hexahedral meshing of non-linear volumes using Voronoi faces and edges. *International Journal for Numerical Methods in Engineering*, 49:329–351, 2000.
32. W.R. Quadros, K. Ramaswami, F.B. Prinz, and B. Gurumoorthy. Laytracks: a new approach to automated geometry adaptive quadrilateral mesh generation using medial axis transform. *International journal for numerical methods in engineering*, 61(2):209–237, 2004.
33. S. Yamakawa and K. Shimada. Fully-automated hex-dominant mesh generation with directionality control via packing rectangular solid cells. *International journal for numerical methods in engineering*, 57(15):2099–2129, 2003.
34. V. Vyas and K. Shimada. Tensor-guided hex-dominant mesh generation with targeted all-hex regions. In *Proceedings of the 18th International Meshing Roundtable*, pages 377–396, 2009.
35. L. Maréchal. Advances in octree-based all-hexahedral mesh generation: handling sharp features. In *Proceedings of the 18th International Meshing Roundtable*, 2009.
36. J.F. Shepherd. Conforming hexahedral mesh generation via geometric capture methods. In *Proceedings of the 18th International Meshing Roundtable*, 2009.
37. S.J. Owen and J.F. Shepherd. Embedding features in a cartesian grid. In *Proceedings of the 18th International Meshing Roundtable*, 2009.
38. Y. Ito, A.M. Shih, and B.K. Soni. Octree-based reasonable-quality hexahedral mesh generation using a new set of refinement templates. *International Journal for Numerical Methods in Engineering*, 77(13):1809–1833, 2009.
39. T.J. Tautges, T. D. Blacker, and S.A. Mitchell. The whisker weaving algorithm: A connectivity-based method for constructing all-hexahedral finite element meshes. *International Journal for Numerical Methods in Engineering*, 39(19):3327–3350, 1996.
40. F. Ledoux and J. C. Weill. An Extension of the Reliable Whisker Weaving Algorithm. In *16th International Meshing Roundtable*, pages 215–232. Springer, 2007.
41. S.A. Mitchell and T.J. Tautges. Pillowing doublets: refining a mesh to ensure that faces share at most one edge. In *4th International Meshing Roundtable*, 1995.
42. J.F. Shepherd, M.W. Dewey, A.C. Woodbury, S.E. Benzley, M.L. Staten, and S.J. Owen. Adaptive mesh coarsening for quadrilateral and hexahedral meshes. *Finite Elements in Analysis and Design*, 46(1-2):17–32, 2010.
43. M.L. Staten, J.F. Shepherd, F. Ledoux, and K. Shimada. Hexahedral mesh matching: Converting non-conforming hexahedral-to-hexahedral interfaces into conforming interfaces. *International Journal for Numerical Methods in Engineering*, 82(12):1475–1509, 2010.
44. J.A. Sethian. *Level set methods and fast marching methods*. Cambridge university press Cambridge, 1999.
45. J. Sarrate and A. Huerta. Efficient unstructured quadrilateral mesh generation. *International journal for numerical methods in engineering*, 49(10):1327–1350, 2000.
46. J. Sarrate and A. Huerta. Automatic mesh generation of nonstructured quadrilateral meshes over curved surfaces in r3. In *3th ECCOMAS conference*, 2000.
47. P. M. Knupp. A method for hexahedral mesh shape optimization. *International journal for numerical methods in engineering*, 58(2):319–332, 2003.
48. P. M. Knupp. Hexahedral and tetrahedral mesh untangling. *Engineering with Computers*, 17(3):261–268, 2001.
49. J. Carreras. Refinament conforme per malles de quadrilàters i hexàedres. Master's thesis, Facultat de Matemàtiques i Estadística. Universitat Politècnica de Catalunya, 2008.
50. J.F. Thompson. *Handbook of Grid Generation*. CRC Press, 1999.
51. X. Roca, J. Sarrate, and E. Ruiz-Gironés. A graphical modeling and mesh generation environment for simulations based on boundary representation data. In *Congresso de Métodos Numéricos em Engenharia*, 2007.
52. X. Roca, E. Ruiz-Gironés, and J. Sarrate. ez4u. mesh generation environment. `http://www-lacan.upc.edu/ez4u.htm`, 2010.