Local Dual Contributions on Simplices: A Tool for Block Meshing[⋆]

X. Roca and J. Sarrate

Laboratori de Càlcul Numèric (LaCàN)
Departament de Matemàtica Aplicada III
Universitat Politècnica de Catalunya
Jordi Girona 1-3, E-08034 Barcelona, Spain
{xevi.roca,jose.sarrate}@upc.edu
http://www-lacan.upc.edu

Summary. Our final goal is to automatically generate a block decomposition of a given domain without previously meshing its boundary. To this end, we propose to obtain directly a valid dual arrangement that leads to a block mesh. In particular, we introduce a tool based on the new concept of local dual contributions. That is, given a domain we first generate a reference mesh composed by simplices, triangles in 2D and tetrahedra in 3D. Then, we add local dual contributions in the elements of a reference mesh to describe a valid dual arrangement. These local dual contributions are added according to a set of hierarchical rules to ensure the correct matching of adjacent contributions. The first implementation of the tool has been successfully applied to the block decomposition of several geometries, ranging from convex and non-convex domains to geometries with holes. Further research is under way in order to extend the applicability of the presented tool to more complicated geometries.

Keywords: Mesh generation, dual, block decomposition, hexahedra, quadrilaterals.

1 Introduction

The finite element method and finite volume method are widely used to solve problems in applied sciences and engineering. Both techniques require a valid discretization of the problem geometry. There exist several mature techniques [1, 2] to obtain these discretizations. However, meshes composed by block elements, quadrilateral (2D) or hexahedra (3D), are preferred for specific problems [3, 4]. Therefore, the solution of these problems demands a tool that automatically generates block elements, see references [5, 6] for a survey of available techniques. On the one hand, several techniques attempt to generate this kind of mesh by directly generating hexahedral elements. On the other hand, the nature of the hexahedral meshes have induced to consider the dual problem [7, 8, 9, 10].

This work is devoted to obtaining a valid topological block decomposition of a given domain without a previous discretization of the boundary. We propose an algorithm that explicitly inserts descriptions of the dual curves (2D) or the dual

[⋆] This work was partially sponsored by the Spanish *Ministerio de Educación y Ciencia* under grants DPI2007-62395 and BIA2007-66965.

surfaces (3D) and handles their intersections. Finally, the primal mesh is obtained from this dual construction. Analogous algorithms have been previously sketched in the literature. In particular, Murdoch et al. [11] briefly introduce the *Twist Plane Insertion* algorithm, and Calvo [12] speculates about the *free stroking*[1] method. In both works such an algorithm is not detailed because there is not a procedure that manages the insertion of dual curves (2D) or surfaces (3D), and that properly describes the intersections. To fill this gap, we introduce here the new concept of local dual contributions which allow to obtain a discrete representation of the dual arrangement. In order to ensure the correct matching of adjacent contributions we present a set of hierarchical rules. The concept of local dual contribution and this set of hierarchical matching rules are the main contribution of our work.

2 Background

2.1 Block Meshes

A mesh in $\mathbb{R}^n$, for our applications n is either 2 or 3, is composed by d-dimensional entities where $d = 0, \ldots, n$. An entity of maximum dimension, i.e. a n-dimensional entity, is also referred as **element**. From 0-dimensional to 3-dimensional entities we use the following notation: **node**, **edge**, **face** and **cell**.

A mesh entity that is topologically equivalent (homeomorph) to a d-dimensional unit interval, i.e. $[0, 1]^d$, is called a **block**. Therefore a mesh composed by blocks is called a **block mesh**. This notation allows to denote to quadrilateral and hexahedral entities with the same term. Hence, quadrilateral and hexahedral meshes are examples of block meshes.

An $(n - d)$-dimensional entity, i.e. an entity of co-dimension d, is referred as a ***d*-co-entity**, where d is an integer value such that $0 \leq d \leq n$. For instance, an element of a mesh is also a **co-node**, and a face in a 3-dimensional mesh is also a **co-edge**. Table 1 shows the correspondences between entities and their associated co-entities for the 2D and 3D cases.

Table 1. Correspondence between d-dimensional entities and their associated co-entities for the 2D (left) and 3D (right) cases

d-entity	d	co-entity	co-dimension
Node	0	co-face	2
Edge	1	co-edge	1
Face	2	co-node	0
	–	–	–

d-entity	d	co-entity	co-dimension
Node	0	co-cell	3
Edge	1	co-face	2
Face	2	co-edge	1
Cell	3	co-node	0

2.2 Dual of a Block Mesh

The concept of mesh dual has been widely used in theoretical [13, 14] and practical [7, 8, 9, 10] works on block mesh generation. Specifically, Murdoch et al.

[1] Named "libre trazado" in the original work in Spanish.

Table 2. Primal entities for a block mesh and their associated dual entities for a quadrilateral and a hexahedral mesh, respectively

Primal	d	Dual	2D dual	3D dual
Node	0	$\mathcal{D}$-co-node	$\mathcal{D}$-face	$\mathcal{D}$-cell
Edge	1	$\mathcal{D}$-co-edge	$\mathcal{D}$-edge	$\mathcal{D}$-face
Quadrilateral	2	$\mathcal{D}$-co-face	$\mathcal{D}$-node	$\mathcal{D}$-edge
Hexahedron	3	$\mathcal{D}$-co-cell	–	$\mathcal{D}$-node

[11], Calvo [12], Thurston [13], and Mitchell [14] extend the concept of mesh dual and present it as an arrangement of intersecting co-curves that bisect blocks in each direction. In 2D, they consider the dual of a quadrilateral mesh defined by a set of curves that bisect the edges of the elements and that intersect in the center of the quadrilaterals. In 3D, the dual of a hexahedral mesh is presented as an arrangement of surfaces that bisect the faces of the hexahedra and that intersect in the center of each hexahedron. We denote this extended view of the dual, referred as Spatial Twist Continuum (STC) in [11], as the **$\mathcal{D}$-space** and one of its composing entities as a **$\mathcal{D}$-co-curve**. That is, in 2D the $\mathcal{D}$-space is an arrangement of **$\mathcal{D}$-curves**, and in 3D is an arrangement of **$\mathcal{D}$-surfaces**. Notice that not all the dual arrangements lead to a valid block mesh. Specifically, Mitchell presents in [14] the conditions that have to be satisfied by a dual arrangement in order to obtain a block mesh from it.

The remaining dual entities of a mesh are obtained by substituting each primal entity by its co-entity. That is, given a d-dimensional primal entity we have an associated dual entity of dimension $n - d$. We assume that dual entities are also in the $\mathcal{D}$-space and we denote them by **$\mathcal{D}$-entities**. Thus, a primal node is substituted by a **$\mathcal{D}$-element**, and a primal element by a **$\mathcal{D}$-node**. Table 2 presents the association between primal entities and their dual co-entities and entities for 2-dimensional and 3-dimensional meshes. For instance, the dual of an edge is always a **$\mathcal{D}$-co-edge** that corresponds to a **$\mathcal{D}$-edge** for the dual of a quadrilateral mesh and to a **$\mathcal{D}$-face** for the dual of a hexahedral mesh.

The $\mathcal{D}$-space is in fact a description of the inherent constraints of the block meshing problem. Note that each co-edge of a block element has an opposed co-edge, and both are connected by $(n - 1)$ $\mathcal{D}$-co-curves. In 2D, each edge has an opposed edge and both are connected by a piece of a $\mathcal{D}$-curve. Along one of these $\mathcal{D}$-curves we have an associated row of quadrilateral elements. In 3D, each hexahedron face is bisected by two $\mathcal{D}$-surfaces that connect the face with an opposite face. The intersection of these $\mathcal{D}$-surfaces define a $\mathcal{D}$-curve associated to a stack of primal hexahedra. In addition, each $\mathcal{D}$-surface determines a layer of primal hexahedra. Note that each $\mathcal{D}$-co-curve is composed by several $\mathcal{D}$-co-edges separated by $\mathcal{D}$-co-faces. Therefore in 2D the $\mathcal{D}$-curves are composed by $\mathcal{D}$-edges separated by $\mathcal{D}$-nodes, and in 3D the $\mathcal{D}$-surfaces are composed by $\mathcal{D}$-faces separated by $\mathcal{D}$-edges.

2.3 Local Dual Contributions

The basic idea of the proposed algorithm is to generate a valid topological block decomposition of a given domain without a previous discretization of the boundary. Our algorithm explicitly inserts descriptions of the $\mathcal{D}$-co-curves in the $\mathcal{D}$-space. Finally, the primal mesh is obtained from this dual construction.

The crucial step is the development of an automatic tool that allows the construction, manipulation and management of $\mathcal{D}$-co-curves. Hence, we propose to create a discrete representation of the $\mathcal{D}$-space referred as **$\overline{\mathcal{D}}$-space**, see Figure 1. To this end, we first create a simple discretization of the domain called **reference mesh**. The proposed algorithm adds discretized versions of the $\mathcal{D}$-co-curves, the **$\overline{\mathcal{D}}$-co-curves**, in the elements of the reference mesh. Each one of the $\overline{\mathcal{D}}$-co-curves is composed by several discretized $\mathcal{D}$-co-edges, denoted by **$\overline{\mathcal{D}}$-co-edges**. We impose that the intersections between different $\overline{\mathcal{D}}$-co-curves are captured by the boundaries of the $\overline{\mathcal{D}}$-co-edges, the **$\overline{\mathcal{D}}$-co-faces**. That is, we consider that our discretized version of the $\mathcal{D}$-co-curves, the $\overline{\mathcal{D}}$-co-curves, define a non-manifold $(n-1)$-dimensional mesh. The intersections between $\overline{\mathcal{D}}$-co-curves and the regions bounded by this arrangement define the $\overline{\mathcal{D}}$-space.

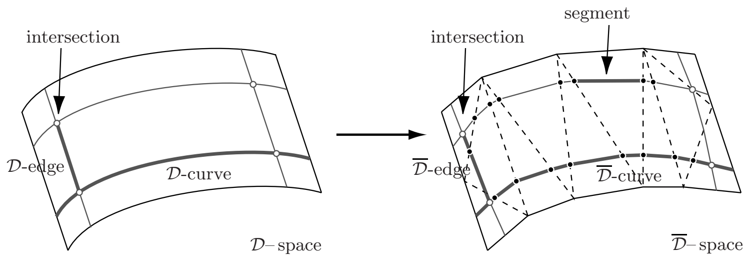
intersection
$\mathcal{D}$-edge
$\mathcal{D}$-curve
$\mathcal{D}-$ space
segment
intersection
$\overline{\mathcal{D}}$-edge
$\overline{\mathcal{D}}$-curve
$\overline{\mathcal{D}}-$ space

Fig. 1. $\mathcal{D}$-space for a 2D domain and its discretized version, the $\overline{\mathcal{D}}$-space

In our discrete approach the domain is substituted by a reference mesh of simplices. We use the simplices of the reference mesh as containers of linear entities of co-dimension one. We propose to describe the $\overline{\mathcal{D}}$-co-curves and their intersections in the $\overline{\mathcal{D}}$-space as the composition of several co-edges, the **local dual contributions**. Thus, for 2D applications we store in the triangles the segments that define the $\overline{\mathcal{D}}$-curves. In Figure 1 we present the $\overline{\mathcal{D}}$-space for a 2D domain. Note that a $\overline{\mathcal{D}}$-curve is composed by several $\overline{\mathcal{D}}$-edges (the piece-wise curves between two white circles). Moreover, each $\overline{\mathcal{D}}$-edge is composed by one or several segments (the straight lines between two black circles) that determine a local contribution to the dual of a triangle of the reference mesh. Analogously, for 3D problems we store in the tetrahedra the planar faces that compose the $\overline{\mathcal{D}}$-surfaces. We remark that each reference element can store none, one or several local dual contributions that can intersect or not between them.

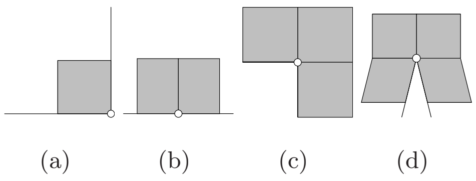

Fig. 2. Vertex classification for 2D domains according to their ideal primal mesh close to the boundary: **(a)** *end*, **(b)** *side*, **(c)** *corner*, and **(d)** *reversal*

2.4 Features

We consider that a domain is described by the following types of entities: **vertex**, **curve**, **surface** and **volume**. Some of the special characteristics of the boundary of the domain, the features, have to be explicitly preserved by the final block mesh. To this end, in this section we classify the features according to the *ideal mesh* around each entity. Note that the name election is based on *Paving* [15] and *Submapping* [16] notation. For a 2D mesh the vertices are classified as, see Figure 2: *i)* ***side***, the ideal mesh is composed by two quadrilateral elements sharing the vertex. In addition, these quadrilaterals have only one edge on the boundary; *ii)* ***end***, the ideal mesh is composed by one quadrilateral element touching the vertex and with only two edges on the boundary; *iii)* ***corner***, the ideal mesh has three quadrilateral elements sharing the vertex. Two elements have only one edge on the boundary, and the middle element has four inner edges; and *iv)* ***reversal***, the ideal mesh has four quadrilateral elements touching the vertex. Two elements have only one edge on the boundary, and the two middle elements have four inner edges.

A vertex is of type ***feature*** if it is not of type ***side***. For 3D applications we have to classify curves and vertices. First, a piece of a curve can be classified as: *i)* ***side***, the ideal mesh is composed by two hexahedral elements sharing the curve and each one with one face on the boundary; *ii)* ***end***, its ideal mesh consists in one hexahedron touching the curve and with two faces on the boundary; *iii)* ***corner***, the ideal mesh has three hexahedral elements incident to the curve. Two elements have only one face on the boundary, and the middle element has six inner faces; and *iv)* ***reversal***, the ideal mesh has four hexahedra touching

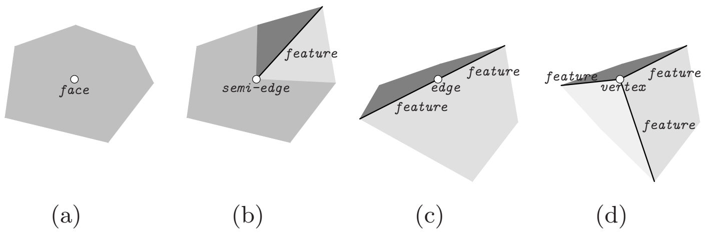

Fig. 3. A sample for each vertex type: **(a)** *face* vertex; **(b)** *semi-edge* vertex; **(c)** *edge* vertex; and **(d)** *vertex* type

the curve. Two elements have only one face on the boundary, and the two middle elements have six inner faces.

A curve is of type `feature` if it is not of type `side`. We determine the classification of the vertices according to the classification of their adjacent curves, see Figure 3: *i)* `face`, there is no adjacent curve of type `feature`; *ii)* `semi-edge`, one adjacent curve is of type `feature`; *iii)* `edge`, two adjacent curves are of type `feature`; and *iv)* `vertex`, more than two adjacent curves are of type `feature`.

3 Constructing the Dual

3.1 Requirements and Aim of the Algorithm

Any block mesh has to fullfill several topological, geometrical and qualitative requirements in order to be used in a numerical simulation. A detailed exposition of these conditions for block meshes can be found in [5, 6]. For the purposes of this work we highlight two of them:

- **Geometric matching.** The block mesh has to reproduce the geometric features of the domain. Moreover, the mesh has to be adapted to each kind of feature. Hence, in Section 2.4 we have defined the vertices (2D) and curves (3D) features according to the ideal mesh around them. For instance, Figure 2 and Figure 4 present the desired primal and dual configuration for each kind of vertex.
- **Boundary sensitivity.** The mesh has to present layers of blocks that follow the boundary of the domain. This condition can be expressed in the $\mathcal{D}$-space, where the layers of blocks are substituted by $\mathcal{D}$-co-curves. Specifically, we have to generate a series of $\mathcal{D}$-co-curves by shrinking the boundary of the domain inwards the domain.

The proposed algorithm is based on the following two paradigms:

- **Block decomposition oriented mesher.** One of the most reliable approaches to block meshing is to decompose the domain in several coarse blocks. The final mesh is obtained by meshing each block and verifying the mesh compatibilities between blocks. Several software packages implement this paradigm [17, 18] providing a set of tools in order to define, by hand, a block topology decomposition of the domain. The user is responsible for determining the topological

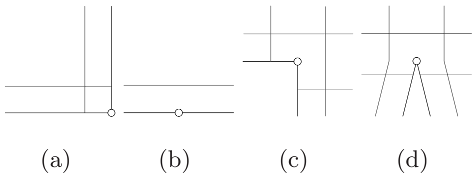

Fig. 4. Ideal mesh dual close to boundary features of a 2D domain: **(a)** `end`, **(b)** `side`, **(c)** `corner`, and **(d)** `reversal`

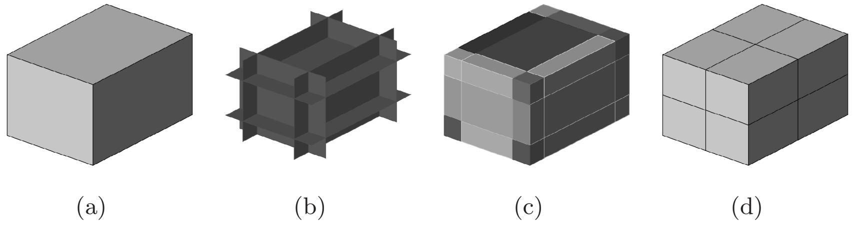

Fig. 5. Block meshing steps in 3D: **(a)** obtain the domain; **(b)** add $\mathcal{D}$-surfaces; **(c)** create $\overline{\mathcal{D}}$-cells; and **(d)** block mesh

relation of the blocks but is allowed to loosely dispose the vertices of the blocks. Then the package is able to adapt the mesh to the domain and to untangle the edges of the blocks by relaxing node location. Therefore, our goal is to obtain a valid coarse block decomposition of the domain.

- **Unconstrained boundary mesher.** A previous discretization of the boundary adds additional constraints that are difficult to fulfill, in particular for hexahedral meshes. Several authors [11, 12] have suggested to develop algorithms in order to mesh domains without a prescribed boundary mesh. In particular, Staten et al. [19] fully develop this idea. In our work we also follow this approach and we propose an algorithm that directly generates the dual of a block mesh without a previous discretization of the boundary.

3.2 Algorithm Proposal

Taking into account the requirements of a valid dual [14] and the requirements of the block mesh, Section 3.1, we propose an algorithm that explicitly inserts descriptions of the $\mathcal{D}$-co-curves in the $\mathcal{D}$-space. Our algorithm is structured in four steps, see Figure 5 for a 3D graphical description.

The main task of the proposed algorithm is to build up a representation of the $\mathcal{D}$-co-curves. To this end, as we mentioned in Section 2.3, we propose to consider a discrete representation of the $\mathcal{D}$-space, the $\overline{\mathcal{D}}$-space. In the following we present an algorithm and we describe the second and the third step:

Algorithm 1. $\overline{\mathcal{D}}$-space block meshing algorithm

Reference mesh. We obtain a Constrained Delaunay Triangulation (**CDT**) of the domain. In our implementation we have used TetGen [20].
Add $\overline{\mathcal{D}}$-co-curves. Create an arrangement of intersecting $\overline{\mathcal{D}}$-co-curves composed by several $\overline{\mathcal{D}}$-co-edges.
Create $\overline{\mathcal{D}}$-co-nodes. Cap the $\overline{\mathcal{D}}$-co-edges with the boundary of the reference mesh. Note that the boundary of the reference mesh is composed by co-edges. We obtain a set of bounded regions, the $\overline{\mathcal{D}}$-co-nodes, sharing at most a common $\overline{\mathcal{D}}$-co-edge.
Block mesh. Once we have the $\overline{\mathcal{D}}$-space, a discretized representation of the $\mathcal{D}$-space, we obtain the final block mesh as the dual of the dual.

Add $\overline{\mathcal{D}}$-co-curves. Each $\overline{\mathcal{D}}$-co-curve is a mesh composed by co-edge elements. We modify the reference mesh in order to make it compatible with the added co-edges. Specifically, we split the elements of the reference mesh to match them with the inserted co-edges. In 2D, the discretized versions of $\mathcal{D}$-co-curves, the $\overline{\mathcal{D}}$-curves, are composed by segments that intersect in a set of shared nodes, the $\overline{\mathcal{D}}$-nodes. The $\overline{\mathcal{D}}$-edges are delimited by the intersections of the $\overline{\mathcal{D}}$-curves. We split the reference mesh elements to match these new segments, see Figure 1. In 3D, the discretized versions of the $\mathcal{D}$-co-curves, the $\overline{\mathcal{D}}$-surfaces, are represented by triangular meshes that can intersect in two possible configurations: *i)* two triangular meshes can intersect in a set of shared edges that compose the discretized versions of the $\mathcal{D}$-edges and $\mathcal{D}$-curves, denoted by $\overline{\mathcal{D}}$-edges and $\overline{\mathcal{D}}$-curves; and *ii)* three triangular meshes can intersect in a shared node which corresponds to a $\overline{\mathcal{D}}$-node. The intersections of the $\overline{\mathcal{D}}$-surfaces bound the discretized versions of the $\mathcal{D}$-faces, the $\overline{\mathcal{D}}$-faces. We split the tetrahedra of the reference mesh obtaining new entities that match properly with the additional triangles that conform $\mathcal{D}$-surfaces.

Create $\overline{\mathcal{D}}$-co-nodes. Taking into account that we have split the reference mesh to add the $\overline{\mathcal{D}}$-co-curves, we obtain a conformal n-dimensional mesh of the $\mathcal{D}$-space, the $\overline{\mathcal{D}}$-space. We obtain a set of regions, the **$\overline{\mathcal{D}}$-co-nodes**, bounded by the $\overline{\mathcal{D}}$-co-edges and the boundary co-edges. In 2D, we obtain a set of areas, the $\overline{\mathcal{D}}$-faces, composed by faces and delimited by the edges that compose the $\overline{\mathcal{D}}$-edges, and the split boundary of the reference mesh. In 3D, we obtain several volumes, the $\overline{\mathcal{D}}$-cells, composed by cells and delimited by the composing faces of the $\overline{\mathcal{D}}$-surfaces, and the split surface mesh of the boundary of the reference mesh.

4 Adding Local Dual Contributions

4.1 Reference Element Contributions

We have outlined an algorithm to obtain a block mesh by means of first generating a discretized version of the dual, see Section 3.2. In the second step of the algorithm, **add $\overline{\mathcal{D}}$-co-edges**, we need a tool for describing the $\overline{\mathcal{D}}$-co-edges and their intersections. To this end, we have introduced the concept of local dual contributions in Section 2.3. Recall that we use a reference mesh of simplices as a discrete representation of the geometry and that local dual contributions are added in the elements of the reference mesh. In this section we detail the geometrical definition for each type of local dual contribution.

In 2D, given a triangle of nodes $\{n_1, n_2, n_3\}$ in a reference mesh, we consider two types of local dual contributions, see Figure 6:

- `edge`. Given an edge e_i, we consider the opposite node n_i. Thus, e_i is determined by the nodes n_{i_1} and n_{i_2}, where $i_1 = i+1 \pmod 3$ and $i_2 = i+2 \pmod 3$. This type of contribution is determined by the segment $\overline{r_{i_1} r_{i_2}}$, where $r_{i_1} := \frac{1}{4}n_i + \frac{3}{4}n_{i_1}$ and $r_{i_2} := \frac{1}{4}n_i + \frac{3}{4}n_{i_2}$.

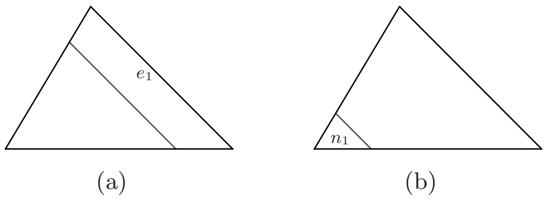

Fig. 6. Local dual contributions on a reference mesh triangle. **(a)** *edge* contribution; and **(b)** *node* contribution.

- *node*. Given the node n_i it is the contribution defined by the segment $\overline{q_{i_1} q_{i_2}}$, where $i_1 = i + 1 (\bmod 3)$ and $i_2 = i + 2 \, (\bmod 3)$, $q_{i_1} := \frac{3}{4} n_i + \frac{1}{4} n_{i_1}$ and $q_{i_2} := \frac{3}{4} n_i + \frac{1}{4} n_{i_2}$.

Note that the local dual contributions intersect with the edges of the reference mesh at distances $\frac{1}{4}$ and $\frac{3}{4}$ of the edge length. On the one hand, this selection ensures that the local dual contributions are compatible between adjacent elements. On the other hand, the intersections between local dual contributions have the proper multiplicity inside the reference elements.

In 3D, given a tetrahedron of nodes $\{n_1, n_2, n_3, n_4\}$ in a reference mesh, we consider three types of local dual contributions, see Figure 7:

- *face*. For a reference tetrahedron we consider a face f of nodes $\{n_{i_1}, n_{i_2}, n_{i_3}\}$ and opposite node n_i. The contribution of this face is defined by the triangle $\{q_{i_1}, q_{i_2}, q_{i_3}\}$ where $q_{i_j} := \frac{1}{5} n_i + \frac{4}{5} n_{i_j}$ for $j = 1, \ldots, 3$.
- *edge*. Consider an edge e delimited by the nodes n_{i_1} and n_{i_2}, and being n_{i_3} and n_{i_4} its opposite nodes. The contribution of e in the tetrahedron is defined by the quadrilateral $\{r_1, r_2, r_3, r_4\}$ with $r_1 := \frac{4}{5} n_{i_1} + \frac{1}{5} n_{i_4}$, $r_2 := \frac{4}{5} n_{i_1} + \frac{1}{5} n_{i_3}$, $r_3 := \frac{4}{5} n_{i_2} + \frac{1}{5} n_{i_3}$ and $r_4 := \frac{4}{5} n_{i_2} + \frac{1}{5} n_{i_4}$.
- *node*. The contribution of a node n_i with opposite nodes $\{n_{i_1}, n_{i_2}, n_{i_3}\}$ is defined by the triangle $\{s_{i_1}, s_{i_2}, s_{i_3}\}$, where $s_{i_j} := \frac{4}{5} n_i + \frac{1}{5} n_{i_j}$ for $j = 1, \ldots, 3$.

For 3D problems, the local dual contributions intersect with the edges of the reference mesh at distances $\frac{1}{5}$ and $\frac{4}{5}$ of the edge length. As in the 2D case, this selection ensures compatibility between neighboring elements and the correct multiplicity of the intersections inside the reference elements.

We propose to use local dual contributions to describe arrangements and intersections of $\overline{\mathcal{D}}$-co-curves. According to Section 3.2, local dual contributions have to define a non-manifold $(n-1)$-dimensional mesh with conformal elements. To capture all the possible intersections with other local dual contributions and obtain the conformal mesh, we decompose each type of local dual contribution in several simplices of co-dimension one. For instance, given a 3D reference mesh we consider the following decompositions for each local dual contribution type, see Figure 8:

- *face*. Given a face contribution defined by the nodes $\{q_1, q_2, q_3\}$, we consider the points $q_{121} := \frac{3}{4} q_1 + \frac{1}{4} q_2$, $q_{122} := \frac{1}{4} q_1 + \frac{3}{4} q_2$, $q_{231} := \frac{3}{4} q_2 + \frac{1}{4} q_3$, $q_{232} := \frac{1}{4} q_2 + \frac{3}{4} q_3$, $q_{311} := \frac{3}{4} q_3 + \frac{1}{4} q_1$, $q_{312} := \frac{1}{4} q_3 + \frac{3}{4} q_1$, $p_1 := \frac{2}{4} q_1 + \frac{1}{4}(q_2 + q_3)$,

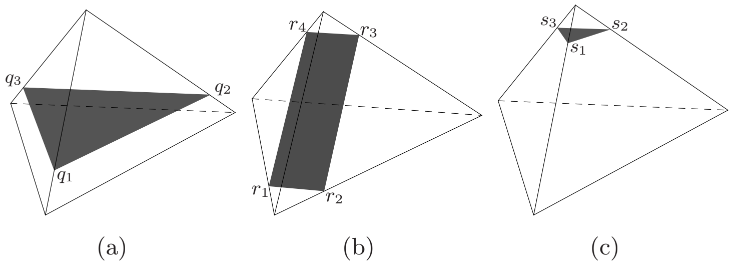

Fig. 7. A sample for each type of local dual contribution: **(a)** contribution of the base face; **(b)** contribution of the frontal edge; and **(c)** contribution of the top node

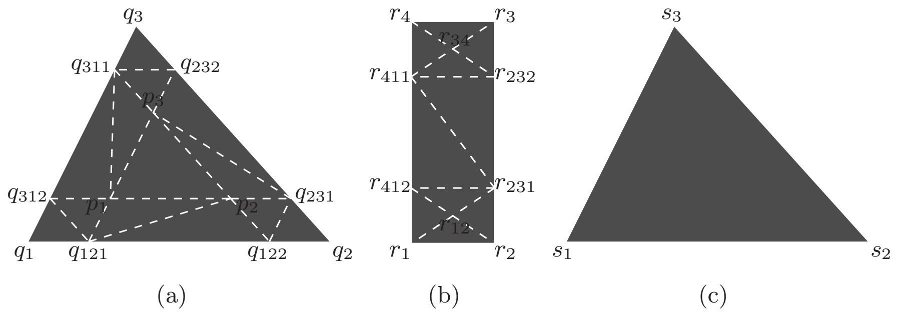

Fig. 8. Associated triangulation for each type of local contribution: **(a)** `face` contribution; **(b)** `edge` contribution; and **(c)** `node` contribution

$p_2 := \frac{2}{4}q_2 + \frac{1}{4}(q_1 + q_3)$, $p_3 := \frac{2}{4}q_3 + \frac{1}{4}(q_1 + q_2)$ and the decomposition of the contribution in the triangles $\{q_1, q_{121}, q_{312}\}$, $\{q_2, q_{231}, q_{122}\}$, $\{q_3, q_{311}, q_{232}\}$, $\{q_{121}, p_1, q_{312}\}$, $\{q_{231}, p_2, q_{122}\}$, $\{q_{311}, p_3, q_{232}\}$, $\{q_{121}, q_{122}, p_2\}$, $\{p_1, q_{121}, p_2\}$, $\{q_{231}, q_{232}, p_3\}$, $\{p_2, q_{231}, p_3\}$, $\{q_{311}, q_{312}, p_1\}$, $\{p_3, q_{311}, p_1\}$ and $\{p_1, p_2, p_3\}$.

- `edge`. Given an edge contribution defined by the nodes $\{r_1, r_2, r_3, r_4\}$, we consider the points $r_{231} := \frac{3}{4}r_2 + \frac{1}{4}r_3$, $r_{232} := \frac{1}{4}r_2 + \frac{3}{4}r_3$, $r_{411} := \frac{3}{4}r_4 + \frac{1}{4}r_1$, $r_{412} := \frac{1}{4}r_4 + \frac{3}{4}r_1$, $r_{12} := \frac{1}{4}(r_1 + r_2 + r_{231} + r_{412})$, $r_{34} := \frac{1}{4}(r_3 + r_4 + r_{411} + r_{232})$ and the decomposition of the contribution in the triangles $\{r_1, r_2, r_{12}\}$, $\{r_2, r_{231}, r_{12}\}$, $\{r_{231}, r_{412}, r_{12}\}$, $\{r_{412}, r_1, r_{12}\}$, $\{r_{411}, r_{231}, r_{232}\}$, $\{r_{412}, r_{232}, r_{411}\}$, $\{r_3, r_4, r_{34}\}$, $\{r_4, r_{411}, r_{34}\}$, $\{r_{411}, r_{232}, r_{34}\}$, and $\{r_{232}, r_3, r_{34}\}$.
- `node`. A node contribution defined by the nodes $\{s_1, s_2, s_3\}$ is not decomposed because it does not have inner intersections with other contributions.

4.2 Hard and Soft Contributions

In order to improve the robustness of the proposed algorithm we require that given two different reference meshes of a single domain the algorithm generates the same dual. For instance, given the reference meshes of the rectangular domain presented in Figures 9(a) and 9(b) we require that the algorithm generates the same dual, Figure 9(c), and the same primal, Figure 9(d).

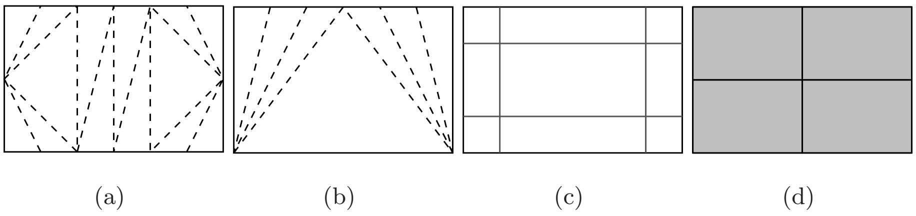

Fig. 9. Decomposition of a rectangular domain. **(a)** First reference mesh; **(b)** second reference mesh; **(c)** ideal mesh dual of the rectangular domain; and **(d)** ideal primal mesh.

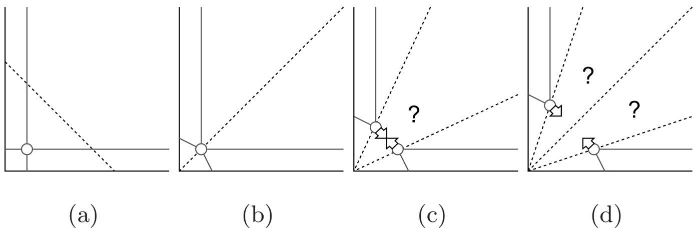

Fig. 10. Local dual contributions for several triangulations around an *end* node: **(a)** one adjacent triangle; **(b)** two adjacent triangles; **(c)** three adjacent triangles; and **(d)** four adjacent triangles. Solid black lines denote the boundary; slashed lines denote the edges of the triangular reference mesh; local dual contributions are denoted by red lines; and the red stroked circles represent the intersection of two local dual contributions.

To this end, consider an *end* vertex of a 2D domain. Depending on the reference mesh this *end* node may belong to different number of elements of the reference mesh, see Figure 10. If the *end* node belongs to one or two elements, see Figure 10(a) and Figure 10(b) respectively, we can add local dual contributions to obtain the ideal $\overline{\mathcal{D}}$-space curves around the *end* node. In the first case two *edge* local dual contributions are needed. In the second case one *edge* and one *node* local dual contributions are needed in each reference triangle adjacent to the vertex.

If the node belongs to more than two triangles, see Figures 10(c) and 10(d), the local dual contribution types presented in Section 4.1 are not sufficient to obtain a valid arrangement of the $\overline{\mathcal{D}}$-curves. The proposed solution for analogous configurations is to virtually collapse the two intersection points into a single virtual point. Then, we obtain the desired two $\overline{\mathcal{D}}$-curves that intersect in the virtual point. To generalize this behavior we classify the local dual contributions in two classes:

- *hard***.** The contribution is respected during the construction of the primal mesh. That is, if at least one of the contributions that separate two $\overline{\mathcal{D}}$-conodes is *hard* we generate a primal edge connecting their associated primal nodes.

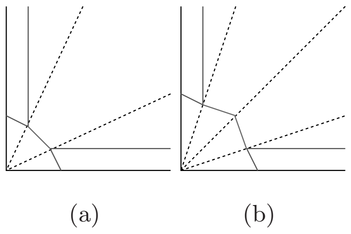
Fig. 11. Local dual contributions for several possible triangulations around an `end` node: **(a)** three adjacent triangles; and **(b)** four adjacent triangles. Red segments represent `hard` contributions and blue segments represent `soft` contributions.

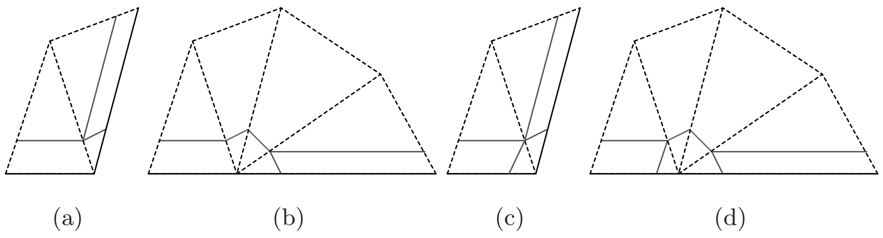
Fig. 12. Examples of local dual contributions in reference triangles. Two non-valid duals: **(a)** three `hard` contributions intersecting at one point; and **(b)** a configuration composed by three `hard` contributions and two `soft` contributions. Two configurations defining a valid dual: **(c)** two `hard` contributions matching at one point; and **(d)** four `hard` contributions and two `soft node` contributions in the middle.

- `soft`**.** The contribution is not respected during the construction of the primal mesh. That is, when all the contributions that separate two $\overline{\mathcal{D}}$-co-nodes are `soft` we do not generate a primal edge connecting their associated primal nodes.

The solution to the problem presented in Figures 10(c) and 10(d) can be interpreted in terms of `hard` and `soft` local dual contributions. On the one hand, the proposed solution is composed by `soft node` contributions of the `end` vertex in each inner triangle. On the other hand, it is composed by two `hard node` contributions of the `end` vertex, and one `edge hard` contribution on each triangle adjacent to the boundary of the reference mesh, see Figure 11. Then, the `soft` contributions will be ignored during the construction of the primal mesh, and this representation of the $\overline{\mathcal{D}}$-space is equivalent to virtually collapsing the `soft` contributions into one virtual point. Note that this virtual point is represented by blue segments in Figure 11.

4.3 Matching Rules

In order to obtain a valid dual we have to ensure that: *i)* local dual contributions compose $\overline{\mathcal{D}}$-co-edges without gaps, and *ii)* the intersections of the $\overline{\mathcal{D}}$-co-edges have the proper multiplicity. For instance, in 2D the multiplicity at the matching points of local dual contributions must be either two or four.

In this sense, the direct application of `hard` and `soft` local dual contributions does not necessarily lead to a valid dual. To illustrate this shortcoming, we consider two configurations that define a non-valid dual and that can be constructed by using `hard` and `soft` local dual contributions. First, Figure 12(a) shows three `hard` local dual contributions sharing an intersection point. Hence, we obtain three $\overline{\mathcal{D}}$-edges and not the required four to obtain a valid dual. Second, Figure 12(b) presents a configuration with `hard` and `soft` local dual contributions. Since `soft` local dual contributions are equivalent to virtually collapse the contributions into one single virtual point, we obtain again three $\overline{\mathcal{D}}$-edges sharing a point. These local dual contributions do not correspond to a valid dual.

To overcome these drawbacks we propose to match the number of contributions from the left and right of a reference mesh co-edge. For instance, in Figure 12(c) the `hard edge` and `node` contributions from the left match with the `hard edge` and `node` contributions from the right, and we obtain a correct piece of $\overline{\mathcal{D}}$-co-edge. Likewise, in Figure 12(d), if we virtually collapse the middle `soft node` contributions we have that `hard` contributions from the left match with the two `hard` contributions from the right. These contributions define the ending parts of four $\overline{\mathcal{D}}$-edges separated by a $\overline{\mathcal{D}}$-node, generated by virtually collapsing the `soft` contributions.

Note that by construction the local dual contributions do not have gaps nor intersections of wrong multiplicity inside the elements of the reference mesh. Thus, after considering that middle `soft` contributions are virtually collapsed, we have to ensure that the `hard` contributions match properly at the boundaries of reference elements. To this end, we propose the following strategies to ensure that the local dual contributions match properly.

- **2D.** The boundaries of the reference triangles are edges. Each edge has two possible intersection points of local dual contributions. We have to check local dual contributions around each one of these points:
  - **Boundary edge.** We allow either no contribution or one incident `hard` contribution from inside the element.
  - **Inner edge.** The number of `hard` contributions from the left have to be equal to the number of `hard` contributions from the right. Thus, the possibilities are: *i)* no incident contributions; *ii)* one from the left and one from the right defining a piece of a $\overline{\mathcal{D}}$-curve; and *iii)* two from the left and two from the right defining four $\overline{\mathcal{D}}$-edges ending at the intersection node of two $\overline{\mathcal{D}}$-curves.
- **3D.** The boundaries of a reference tetrahedron are determined by four triangular faces. Each face has six possible segments where local dual contributions can intersect. These segments are determined by all the possible `face`, `edge`, and `node` contributions inside the tetrahedron. Note that for each intersection segment there are only two possible contributions. Hence, we have to check local dual contributions around each one of these segments:
  - **Boundary face.** We allow either no contribution or one incident `hard` contribution from inside the element.

- **Inner face.** The number of `hard` contributions from the left has to be equal to the number of `hard` contributions from the right. Thus, the possibilities are: *i)* no incident contributions; *ii)* one from the left and one from the right defining a piece of a $\overline{\mathcal{D}}$-surface; and *iii)* two from the left and two from the right defining four $\overline{\mathcal{D}}$-faces ending at the intersection of two $\overline{\mathcal{D}}$-surfaces.

4.4 Implementation

We can add local dual contributions from the composing entities of the reference mesh. In addition, we can use `hard` and `soft` contributions in order to capture the desired dual. But we have to add these local dual contributions taking into account the set of matching rules. To this end, in our implementation we add the contributions using a hierarchical procedure: we start at the highest dimension entities (co-edge contributions) and finish at lowest dimensional entities (`node` contributions). In this scheme, highest dimensional contributions are responsible of determining the main shape of the desired dual, and lower dimensional contributions are responsible of filling the gaps and matching local dual contributions properly. Algorithm 2, details a procedure for 3D applications.

We use this procedure to implement our local dual contributions tool. The last two steps are responsible of obtaining a valid dual once the `face` contributions have been added. These steps are done automatically according to the matching rules and the required ideal mesh around the features. The difficulties reside in the first step where we have to decide which `face` contributions to add to finally obtain the desire $\overline{\mathcal{D}}$-surfaces.

Algorithm 2. Add local dual contributions

Add `face` contributions. We add `hard` face contributions: *i)* to capture one layer of hexahedra that follows the boundary (adding a `face` contribution for each boundary face); *ii)* to split vertices with more than three incident feature curves; and *iii)* to capture ideal mesh close to curve and vertex features.

Add `edge` contributions. Taking into account previously added `face` contributions, we automatically add `edge` contributions according to the matching rules. Thus, we add `edge` contributions: *i)* to cover the gaps between `face` contributions with `soft` contributions; *ii)* to ensure that `face` contributions match with correct multiplicity using `soft` and `hard` contributions; and *iii)* to capture the ideal mesh close to curve and vertex features with `hard` contributions.

Add `node` contributions. Taking into account previously added `face` and `edge` contributions we can still have a non-valid arrangement of $\overline{\mathcal{D}}$-surfaces. To obtain the final set of intersecting $\overline{\mathcal{D}}$-surfaces we add `node` contributions: *i)* to fill all the remaining gaps between `face` and `edge` contributions with `soft` node contributions; *ii)* to match all the `hard` `face` and `edge` contributions and to define valid $\overline{\mathcal{D}}$-surfaces intersections using `soft` and `hard` contributions; and *iii)* to capture the ideal mesh at the vertex features with `hard` contributions.

5 Examples

The aim of the examples presented in this section is to show that the concept of local dual contributions has enough expressivity to capture the dual of the desired block mesh for a wide range of geometries: convex and non-convex domains, with 3 or 4-valenced vertices, and domains with holes.

We use our implementation to decompose these geometries in coarse blocks. It is important to point out that the proposed algorithm focuses on the generation of valid topological block decomposition. Then, the final node location can be improved using a relaxation procedure [17, 18]. In addition, a finer mesh can be obtained by meshing each block separately while keeping the mesh compatibility conditions between them.

For the three first examples, Figure 13 shows the five main steps involved in the block meshing process: *i)* to obtain hte reference mesh, the CDT of tetrahedra; *ii)* to add local dual contributions to define $\overline{\mathcal{D}}$-surfaces and their intersections, where red faces represent the ***hard*** local dual contributions and blue faces represent the ***soft*** local dual contributions; *iii)* to cap the $\overline{\mathcal{D}}$-surfaces and obtain a $\overline{\mathcal{D}}$-space where cells are colored according to the containing $\overline{\mathcal{D}}$-cell; finally, *iv)* and *v)* to obtain the primal nodes (colored as the associated $\overline{\mathcal{D}}$-cells), edges, faces and hexahedra of the block mesh.

In the first example, we decompose in eight blocks a brick object that is tapered in two orthogonal directions, see Figure 13(a). In this example all the vertices are 3-valenced. Thus, it can be meshed with submapping or midpoint subdivision. In fact, the obtained decomposition is equivalent to a coarse mesh that can be obtained by these methods. In this example all the $\overline{\mathcal{D}}$-surfaces follow the boundary of the domain and the possible gaps between contributions are filled with ***soft*** contributions.

For the second example, see Figure 13(b), we consider an extruded pentagon. The final block decomposition is equivalent to the mesh we would obtain using midpoint subdivision. Notice that in this example the submapping method will lead to a non-symmetrical block decomposition of lower quality. The $\overline{\mathcal{D}}$-surfaces are discretized versions of $\mathcal{D}$-surfaces that follow the boundaries of the domain. Close to the top front vertex the $\overline{\mathcal{D}}$-surfaces have a ***soft*** local dual contribution. When the algorithm caps the $\overline{\mathcal{D}}$-surfaces we obtain a region close to the top front vertex colored with cyan. This dual region, $\overline{\mathcal{D}}$-cell, is associated to the top front node of the primal, colored in cyan. Notice that since the local dual contribution is ***soft***, the primal mesh does not show an edge connecting the top front node with the center node of the top of the volume. That is, during the generation of the primal mesh we ignore the ***soft*** contribution.

The third example presents the decomposition of a double pyramid, see Figure 13(c). Note that we can use neither submapping nor midpoint subdivision because all the vertices are 4-valenced. In this case our local dual contributions tool automatically splits the double pyramid in 4 tetrahedra. Moreover, the final mesh is composed by 16 hexahedra that correspond to the decomposition of each one of

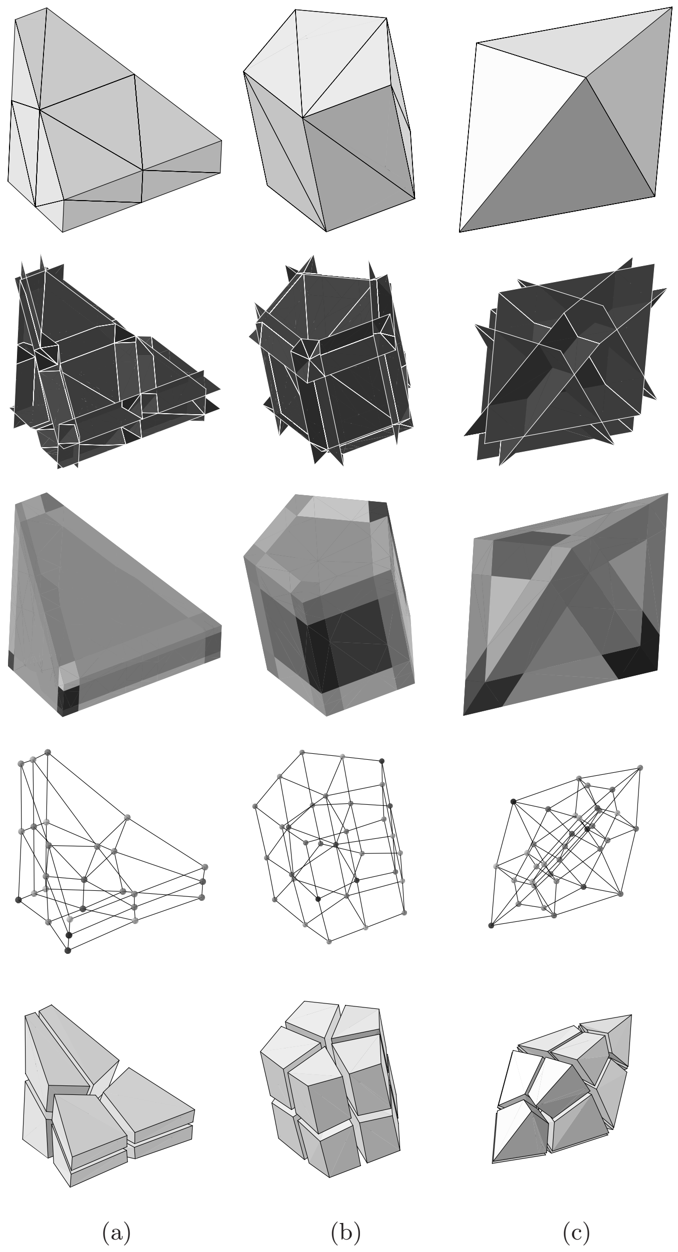

Fig. 13. Block meshing steps for three different domains: **(a)** a domain with 3-valenced vertices; **(b)** an extruded pentagon; and **(c)** a domain with 4-valenced vertices. Rows represent the algorithm steps: *i)* reference mesh; *ii)* $\overline{\mathcal{D}}$-surfaces composed by local dual contributions; *iii)* $\overline{\mathcal{D}}$-space where $\overline{\mathcal{D}}$-cells are colored in base of the containing $\overline{\mathcal{D}}$-cell; *iv)* primal nodes (colors associated to $\overline{\mathcal{D}}$-cells) and edges; and *v)* block mesh.

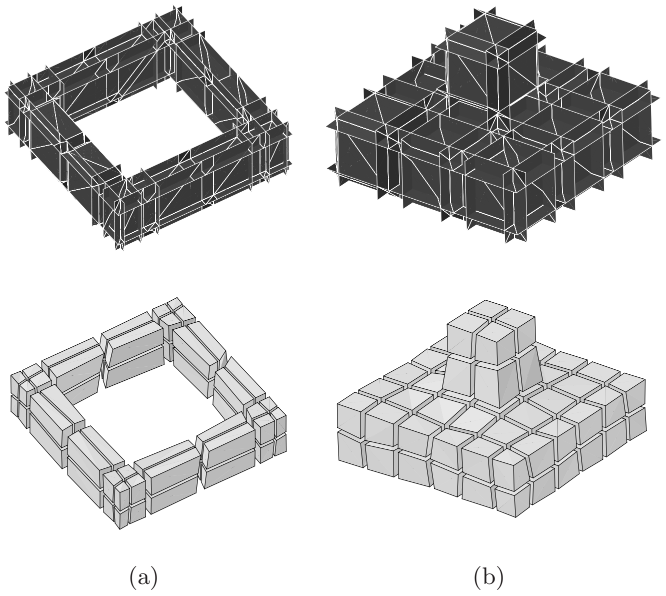

Fig. 14. The $\overline{\mathcal{D}}$-surfaces and the block meshes for **(a)** a domain with a hole defined by four `corner` curves; and **(b)** a convex domain with a 3-valenced protrusion

the 4 tetrahedra in 4 hexahedra. In this case all the contributions are `hard`, and all the inner parallel $\overline{\mathcal{D}}$-surfaces are associated with an inner quadrilateral primal mesh that splits the domain in two parts.

The last two examples, two non-convex domains, are decomposed in a larger number of hexahedra than the three first examples. Therefore, we only show the local dual contributions and the obtained primal meshes, see Figure 14.

In the fourth example we discretize the extrusion of a square-shaped ring, Figure 14(a). This test domain has four `feature` curves of type `corner` (non-convex domain), a hole that cross the whole domain, and all the nodes are 3-valenced. Observe that we obtain ideal primal and dual configurations close to the `corner` curves. In fact, these configurations are the 3D generalization of the ideal configurations presented for `corner` vertices in Figure 4.

The last example is an extruded square with a protrusion on the top, Figure 14(b). All the nodes are 3-valenced and the curves that join the protrusion with the base are of type `corner`. We obtain a coarse decomposition of the domain that could also be obtained with a submapping algorithm. Since we consider the unconstrained boundary mesh approach, we do not have to previously compute a compatible interval assignment of the boundary. The four quadrilaterals that separate the protrusion from the rest of the mesh are associated to two parallel $\overline{\mathcal{D}}$-surfaces.

6 Concluding Remarks and Future Work

In this paper we present a new algorithm to generate a block decomposition of a given domain when there is not a prescribed boundary mesh. To this end, we propose an approach to directly construct a valid dual of the block mesh. The proposed algorithm is composed by two main steps. First, we create a reference mesh from the data of the boundary domain by means of a CDT. Second, we create the dual of the block mesh using the new concept of local dual contributions. These contributions are generated in the elements of the reference mesh. Specifically, we have implemented a tool that automatically adds local dual contributions following a set of hierarchical rules. The proposed implementation ensures that adjacent local contributions match properly. That is, the local dual contributions define a dual of the block mesh with intersections of the proper multiplicity and without gaps.

The main goal of the first implementation of the method is to obtain block meshes and their duals for some simple configurations: convex and non-convex domains defined by planar surfaces. Our current implementation allows to automatically mesh these simple domains. However, we can also mesh some domains with several vertices having a valence greater than three. In all theses cases the developed tool generates the expected mesh.

Practical results suggest that the proposed types of local dual contributions and the set of matching rules incorporate enough expressivity to describe the dual arrangement of a block mesh. However, it is expected that as the tool is applied to more complicated geometries (non-blocky geometries, domains with sharp angles, or assembly models) additional logic might be incorporated.

Note that the proposed tool is under continuous development. In this sense, the current version of this tool has several limitations that should be investigated and solved in the near future. First, we have to analyze the dependence of the algorithm from the reference mesh. For instance, it is not clear how the resolution of the reference mesh will affect the topology of the block decomposition. Second, it has to be investigated how the small perturbations of the boundary will affect the final primal mesh. Third, we have to apply the developed tool to domains described by curved boundaries that include smooth details. Finally, we have to check the robustness of the proposed tool in applications of practical interest.

References

1. Owen, S.J.: A survey fo unstructured mesh generation technology. In: 7th International Meshing Roundtable, pp. 239–267 (1998)
2. Baker, T.J.: Mesh generation: Art or science? Progress in Aerospace Sciences 41(1), 29–63 (2005)
3. Benzley, S., Perry, E., Merkley, K., Clark, B., Sjaardema, G.: A comparison of all-hexahedral and all-tetrahedral finite element meshes for elastic and elasto-plastic analysis. In: 4th International Meshing Roundtable, pp. 179–191 (1995)
4. Cifuentes, A.O., Kalbag, A.: A performance study of tetrahedral and hexahedral elements in 3-d finite element structural analysis. Finite Elements in Analysis and Design 12(3-4), 313–318 (1992)

5. Blacker, T.: Automated conformal hexahedral meshing constraints, challenges and opportunities. Engineering with Computers 17(3), 201–210 (2001)
6. Tautges, T.J.: The generation of hexahedral meshes for assembly geometry: survey and progress. International Journal for Numerical Methods in Engineering 50(12), 2617–2642 (2001)
7. Tautges, T.J., Blacker, T., Mitchell, S.A.: 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)
8. Folwell, N.T., Mitchell, S.A.: Reliable whisker weaving via curve contraction. Engineering with Computers 15(3), 292–302 (1999)
9. Mueller-Hannemann, M.: Hexahedral mesh generation by successive dual cycle elimination. Engineering with Computers 15(3), 269–279 (1999)
10. Calvo, N.A., Idelsohn, S.R.: All-hexahedral element meshing: Generation of the dual mesh by recurrent subdivision. Computer Methods in Applied Mechanics and Engineering 182(3-4), 371–378 (2000)
11. Murdoch, P., Benzley, S., Blacker, T., Mitchell, S.A.: The spatial twist continuum: A connectivity based method for representing all-hexahedral finite element meshes. Finite Elements in Analysis and Design 28(2), 137–149 (1997)
12. Calvo, N.A.: Generación de mallas tridimensionales por métodos duales. PhD thesis, Facultad de Ingeniería y Ciencias Hídricas (2005)
13. Thurston, W.: Hexahedral decomposition of polyhedra, `http://www.ics.uci.edu/~eppstein/gina/Thurston-hexahedra`
14. Mitchell, S.A.: A characterization of the quadrilateral meshes of a surface which admit a compatible hexahedral mesh of enclosed volume. In: Puech, C., Reischuk, R. (eds.) STACS 1996. LNCS, vol. 1046, pp. 465–478. Springer, Heidelberg (1996)
15. Blacker, T., Stephenson, M.B.: Paving: A new approach to automated quadrilateral mesh generation. International Journal for Numerical Methods in Engineering 32(4), 811–847 (1991)
16. Whiteley, M., White, D., Benzley, S., Blacker, T.: Two and three-quarter dimensional meshing facilitators. Engineering with Computers 12, 144–154 (1996)
17. Ansys. Ansys icem cfd, `http://www.ansys.com/products/icemcfd.asp`
18. Program Development Company. Gridpro, `http://www.gridpro.com/`
19. Staten, M.L., Owen, S.J., Blacker, T.: Unconstrained paving and plastering: A new idea for all hexahedral mesh generation. In: 14th International Meshing Roundtable (2005)
20. A quality tetrahedral mesh generator, `http://tetgen.berlios.de`