DUAL-BASED USER-GUIDED HEXAHEDRAL BLOCK GENERATION USING FRAME FIELDS.

Simon Calderan$^{1,2}$ Guillaume Hutzler$^2$ Franck Ledoux$^1$

$^1$CEA, DAM, DIF, F-91297, Arpajon, France. simon.calderan@ocre.cea.fr
$^2$IBISC, University of Evry, Paris-Saclay, France

ABSTRACT

Block structured hexahedral meshing is required for many applications but remains unreachable in an automatic manner for realistic simulation models. In practice, true industrial cases are handled using complex and rich interactive software, and generating a block structure for a mechanical CAD part can require several days to weeks for an expert engineer. For many years now, several scientific works have demonstrated that 3D frame fields are a very powerful tool for hexahedral meshing. These works remain mainly theoretical and their application is limited to simple 3D models. In this paper, we propose to provide the necessary components to build an interactive tool using frame fields. The principle of the approach is to build a valid dual structure made of 3D dual sheets, which are aligned along a 3D frame field, and then to convert this dual structure into its primal hexahedral block structure.

Keywords: blocking, hexahedral mesh generation, frame field, user-guided approach, dual structure

1. INTRODUCTION

For many application fields, such as high deformation mechanics, hydrodynamics or fluid dynamics, hexahedral meshes are preferred over tetrahedral meshes by physics scientific codes and the underlying numerical methods. More specifically, boundary-aligned hexahedral block-structured meshes are required. But such meshes are known to be very hard to create in practice. And despite the active research done in the last three decades, the generation of such meshes in an industrial context remains a manual task that can need several days to weeks depending on the CAD model complexity. It is clearly unbearable in the production life cycle and it is known that generating meshes in general will be keeping to be an important bottleneck in the near future [1].

In the past few years, many research teams working on hexahedral meshes have studied the usage of frame fields to reveal the expected block structure. They proposed different algorithms and heuristics but all of them were only able to get acceptable results on simple CAD models. Complicated ones are only meshed using mixed meshes$^1$ or hex-dominant meshes [2, 3, 4]. We think that frame fields are a relevant and strong tool for the generation of hexahedral meshes but they encounter at least two issues up to now: (1) no heuristics used to generate frame fields guarantees that their structure fits the hexahedral block structure; (2) even if the frame field is generated, the extraction of a block structure is not so automatic and robust as it should be. From this observation, we propose to consider a hybrid approach where the engineer will guide the algorithm in the generation process. Instead of providing a fully automatic procedure, we intend to provide useful tools to interactively generate hexahedral blocks starting from an input frame field. This way, we expect to integrate frame fields technology into industrial tools in a short time and so to contribute to diminish the global time to simulation.

$^1$In 3D, a mixed mesh can contain any type of 3-dimensional cells: tetrahedra, hexahedra, prisms, pyramids and polyhedra.
Figure 1: Starting from a tetrahedral mesh $T$ where a frame is assigned at every node (a), we first select a set of dual sheets (b) that are used to create a block structure (c), which is eventually refined to get a hexahedral mesh with the expected size properties (d).

1.1 Related works

Our approach consists in extracting 3D dual sheets from a frame field. Using frame fields to interactively generate blocks was very recently proposed by [5] in a quite similar way. In their work, the authors provide an interactive environment to design dual sheets starting from boundary loops that the user can define in different manners. We follow the same path but instead of using an indirect approach where the user builds boundary loops using a surface cross field or some surface interactors disconnected from cross fields, we directly build 3D dual sheets. Authors of [6] provide an automatic algorithm based on the same concept, which is to build boundary loops starting from a surface cross field. In both cases, 3D dual sheets are deduced from boundary loops by using an implicit definition of surfaces [5] or a volume partitioning technique based on the min-cut, max-flow algorithm [6].

Using frame field for producing hexahedral meshes has received much interests during the past few years [7, 8, 9, 10]. Frame fields provide directional information in the whole domain to be meshed. It can be used to drive tetrahedral mesh generation before applying a tet-to-hex recombination algorithm to generate mixed meshes, to directly extract the hexahedral block structure [8, 9], or to generate a parameterization that will then give the final mesh [11, 12]. Frame field technology seems to have a high potential to get quality all-hex meshes, but there remains a lot of work to do since the topology of generated frame fields does not always fit the topological structure of hexahedral meshes. As a consequence, frame fields are also used for generating high-quality hex-dominant meshes [2, 4]. In the present work, frame fields are only an input and we do not focus on the way to generate it.

The dual structure of hexahedral meshes has been studied for its peculiar properties [13, 14, 15, 16] that are global to the mesh. Comparing to tetrahedral meshes where it is quite easy to perform local modifications as removing a node or swapping an edge, modifying hexahedral meshes in a safe manner\(^2\) is more global using operations on sheets or chords. The topological structure of hexahedral meshes has so been used to generate hexahedral meshes considering topology first via the Whisker-weaving algorithm [17, 18, 19] or the dual cycle elimination [20, 21]. Either modifying meshes by inserting fundamental sheets to get a block-structure hexahedral mesh [22] or matching two hexahedral meshes along a contact surface [23] to get a full conformal hexahedral mesh for a CAD assembly model has been observed to be useful.

1.2 Method overview

Our method can be seen as similar to [5, 6] on many aspects, but is fundamentally different on one point. We build 3D dual sheets directly and not as enclosed by dual surface loops. Building a 3D sheet is more challenging on many points but it has the benefit of avoiding some side-effects: (1) a dual sheet can be enclosed by several dual boundary loops and finding the set of dual boundary loops that corresponds to the same dual sheet is not an obvious task; (2) it takes advantage of the whole geometrical information carried on the 3D frame fields inside the domain. The approach we propose consists in a pipeline that is made of three main steps (see Figure 1). Starting from a tetrahedral mesh $T$ where a discrete frame field is given on every node of $T$, we adopt the following process:

1. First, we interactively build a set of dual sheets that splits the volume into several dual zones (see Figure 1-b). A dual sheet is built from a picked tetrahedral cell of $t \in T$ and a normal direction, which is one of the three frame directions linearly interpolated at the center of mass of $t$.

2. Second, we extract a block structure from the dual structure previously built (see Figure 1-c). This stage can fail if the dual structure provided

\(^2\)In the meaning that the mesh always remains full-hex during the whole modification process.
the dual structure. But beforehand, we introduce several useful notions in Section 2.

1.3 Main benefits and drawbacks of our method

A main difference between our works and [5, 6, 24] is that we directly build 3D sheets to decompose the geometric domain Ω we work on. It induces a main disadvantage, which is that our approach relies on tracing 3D stream surfaces along a 3D frame field defined in Ω. Tracing such streamlines is more complex than tracing surface loops. It is due to the potential non-integrability of the 3D frame field we start from and the numerical issues that are more difficult to control and handle in a robust programmatic way.

But we believe that the advantages of dealing with 3D sheets deserve to try and get a robust stream surface building process. Only considering surface loops as in [5, 6, 24] imply several restriction and extra process. First, the underlying assumption is that a loop encloses a 3D sheet in the volume, and so corresponds to generating a 3D sheet. This assumption is very restrictive as a 3D sheet can be bounded by several surface loops (depending on the domain genus). A first issue is so to define a 3D sheets as bounded by several loops and not just one. By dealing with 3D sheets directly we avoid the heuristics proposed in [5, 6] for such situations. A second issue is that cross fields does not provide all the required information to build the 3D sheet bounded by one or several surface loops. As a consequence, [5] cut the volume following a constant direction, which is defined by the normal to the surface where the loop is defined (when it is possible), or provides interactive control to the users, who can add control points to deform the 3D sheets. In [6], the 3D surface is built automatically following a 3D field using a min-cut max-flow graph cutting algorithm. A third issue is that those approaches seem not be able to handle self-intersecting loop while a direct 3D approach allows it.

Considering now the full 3d approaches that automatically build a 3D singularity graph and extract a block structure[7, 11, 12, 8, 9, 2, 4], they are all of them limited to simple CAD examples and some CAD features are not possible to be handled (see examples of Fig. 24 and 26 for instance). It is also our case in this paper, but we consider that this work gives some mandatory components to design a new interactive tool with which the engineers we work with will be able to generate full-hex meshes more quickly than with their current tools.

2. BACKGROUND

Our approach relies on two main notions: frame fields and the hexahedral dual structure. In this section, we give the minimum of knowledge that is required for reading this paper. For more information about frame fields, please refer to [10, 12] and for the hexahedral dual structure, take a look at [13, 15].

2.1 Notions of frame fields

For the purpose of this work, we only consider discrete frame fields associated to a tetrahedral mesh. More specifically, given a tetrahedral mesh T that discretizes a CAD model Ω, we consider that a frame field associates a frame to every node of T.

A 3D frame F is defined as a 3-tuple \{u, v, w\}, where u, v and w are three unit 3D vectors such that \( u \cdot v = 0 \) and \( w = u \times v \). In other words, F is defined by three unit vectors, that are orthogonal to each other forming an orthonormal right-handed basis of \( \mathbb{R}^3 \).

3D frames are naturally connected to the structure of hexahedral elements. Considering a hexahedral cell, which has 3 main directions linking its pairwise opposite faces, 24 frames can thus represent it. This set of frames corresponds to the cubical symmetry group \( G \) (any map in SO(3) which maps coordinate axes to coordinate axes) and is invariant under rotations of \( \frac{\pi}{2} \) around one of its three axis and forms an equivalence class.

A discrete frame field can be generated by many methods [9, 10, 12]. To our knowledge all of them try to get a smooth field while preserving the boundary alignment along Ω. As a result, frames assigned to boundary nodes have one of their three directions aligned with the surface normal. Most of tetrahedral cells are called regular and a few ones are called singular. The frame field linearly interpolated in a regular tetrahedron does not contain any singularity, while a singular tetrahedron contains singularities (see Figures 2 and 3). We perform linear interpolation between frames into a tetrahedral cell by using a unit
quaternion representation as done in [8]. In the con-
text of discrete frame fields, we denote SG(T) the set
of singular tetrahedra of T, and we name it singu-
lar graph set. The set SG(T) contains one-direction
stable singular tetrahedra and unstable singular tetra-
hedra. The former are traversed by a single singularity
line (see green tetrahedra on Fig. 3-
a) while the latter
contain a singularity point, i.e. a meeting point be-
tween several singularity lines (see red tetrahedra on
Fig. 3-b).

2.2 Primal and dual hexahedral mesh
Building the dual of a mesh is a quite well-known pro-
cess. To introduce it in the context of this work, we
restrict our presentation to hexahedral meshes that
are 3-dimensional meshes, or 3D meshes for short.
Let us first introduce some terminology. A hexahe-
dral mesh is made of hexahedral elements, which are
3-dimensional cells or 3-cells, quadrilateral faces, or
2-cells, edges, or 1-cells, and nodes, or 0-cells. For
a mesh M, we denote Mi, with i ∈ {0; 3}, the set
containing all the i-cells of M. We only consider con-
formal meshes. In other words, a 3-cell of a mesh
M can only share an i-cell with another 3-cell, with
i ∈ {0, 1, 2}. The dual DM of a mesh M is also a 3D
mesh, where:
- Each i-cell of DM corresponds to a (3 − i)-cell
of M, with i ∈ [0; 3]. For any i-cell c of M, we
denote di−1 the corresponding cell in DM.
- If two 3-cells a3 and b3 of M share a face f2, then
nodes d0a and d0b are linked by edge d1f.

An example of a simple hexahedral mesh M and its
dual mesh DM is given in Figure 4.
This domain decomposition made of dual sheets and dual zones is the dual structure, also noted hex layout in [5].

![Figure 5](image)

**Figure 5:** In (a), the minimal block structure for a portion of sphere; In (b) is shown the dual sheets and the block structure; On (c), the dual zone A is delimited by the red; blue green ad yellow dual sheets; In (d); a single hexahedral block corresponds to a dual node which is the intersection of 3 dual sheets: the green, yellow and red ones.

The notion of singularity graph $SG(T)$ introduced as the set of locations where frame fields are singular in Section 2.1 has also a significant meaning in the block structure, and more widely in hexahedral mesh topology. The singularity graph of a hexahedral mesh, and so a block structure, is the set of inner edges that are not adjacent to four hexahedral cells. In practice, those edges are mainly adjacent to three or five hexahedral cells.

3. **DUAL SHEETS CREATION**

The first stage of our approach consists in creating a set of dual sheets inside our domain $\Omega$. Our input is a tetrahedral mesh $T$ that discretizes $\Omega$ and where frames are assigned at every node. The process we propose is the following one:

1. The user picks a tetrahedral cell $t \in T$. Then we linearly interpolate in $t$ a frame $F_t$ from the frames assigned to the corners of $t$. Note that we allow only regular tetrahedra to be picked, so that this interpolation is well-defined.

2. Then we select one of the 3 directions of $F_t$. Let’s call this direction $d$.

3. We try and extract a dual sheet starting from $t$ with normal direction $d$. This process is described below. It relies on two main concepts: sheet propagation (see Section 3.1) and control filters (see Section 3.2). It can fail if the extracted dual sheet encounters a tetrahedron of $SG(T)$ in an unacceptable configuration.

4. If the sheet extraction succeeds and satisfies the user expectations, the user repeats steps 1 to 3 until getting the desired set of dual sheets.

3.1 **Single sheet propagation**

We start by interactively selecting the input of our dual sheet extraction algorithm, which is summarized by Algorithm 1. Let $t_0 \in T$ be this first tetrahedron. Our extraction method consists in propagating a cutting surface named $S_c$ in $T$ using the frame field $F$. In order to define $S_c$, the user selects one of the 3 directions of the frame $F_0 = (u_0, v_0, w_0)$ defined in $t_0$. Let $d_0$ be this direction vector. We call $E_{t_0}$ the set of triplets $(e, n, \beta)$, where $e$ is an edge of $t_0$ intersected by $S_c$, $n$ a locally normal vector of $S_c$ at the point $p$ on $e$, with $p = (1 - \beta)c_0 + \beta c_1$, where $(c_0, c_1)$ are the end points of edge $e$ and $\beta \in [0, 1]$. Tetrahedra adjacent to an edge of $E_{t_0}$ are the next tetrahedra to be cut. At each propagation step $i$ of the algorithm, they form the wave $W_i$. They will be cut according to the information gathered in $E_{t_0}$. And the process is repeated until reaching an iteration $j$ where $W_j = \emptyset$.

**Initialization of the sheet propagation.** The first step of Algorithm 1 (line 1) consists in creating the initial cutting plane in $t_0$ (see Figure 7). To do this, we define the operation $\text{closest}_\text{component}(F_0, v)$ that finds the closest vector of $v$ among the vectors $(\pm u_0, \pm v_0, \pm w_0)$. We call $c_0$ the center of mass $t_0$, $F_c$ the frame interpolated at $c$ from the frames defined at the nodes of $t_0$ and we note $n_0 = \text{closest}_\text{component}(F_c, d_0)$. We then create the plane $P_0$ going through $c$ and having $n_0$ as a normal vector (see Figure 7(b)). The set $E_{t_0}$ is the set of triplets $(e, n, \beta)$, where: $e$ is an edge of $t_0$ intersected by $P_0$ at point $p_0$ (a red edge on Figure 7(b)); $\beta$ is the barycentric coordinate of $p_0$ along edge $e$; $n$ is equal to $\text{closest}_\text{component}(F_0, n_0)$, where $F_\beta$ is the frame interpolated$^3$ at $p_0$ from the frames defined at the end points of $e$ (points $p_i$ are shown in blue on Fig. 7-b).

**Propagation loop strategy.** Then in order to create the sheet, we propagate the surface $S_c$ in a breadth-first manner. We get all the

$^3$Linear frame interpolation is done using a unit quaternion representation of frames, as done in [9].
Figure 6: Illustration of the sheet creation into a regular field: In the top row, the set of intersected tetrahedra at different loop stages; in the bottom row, a surface-like representation of the advancing dual sheet. In (a), we start from a single tetrahedron and then we advance in (b) and (c) into the domain following the frame field until reaching the boundary domain in all the directions in (d). Color of an element (piece of surface, tetrahedron) is correlated to the wave number.

Algorithm 1: dual sheet creation

Data: \((t_0, d_0)\)

Result: All the tetrahedra intersected by the dual sheet

1. \(\mathcal{E}_{t_0} = \{(e, n, \beta)\} \leftarrow \text{cutFirstTet}(t_0, d_0);\)
2. \(\mathcal{W}_i \leftarrow \text{nexTets}(\mathcal{E}_{t_0}, \{t_0\});\)
3. \(\mathcal{E}_{W_i} \leftarrow \mathcal{E}_{t_0};\)
4. \(\text{done} \leftarrow \text{false};\)
5. while \(!\text{done}\) do
6. \(\mathcal{E}_{W_{i+1}} \leftarrow \emptyset;\)
7. for \(t_k \in \mathcal{W}_i\) do
8. \(\quad \mathcal{E}_{W_{i+1}} \leftarrow \mathcal{E}_{W_{i+1}} \cup \text{cutTet}(t_k, \mathcal{E}_{W_i});\)
9. end
10. \(\mathcal{E}_{W_{i+1}} \leftarrow \text{applyControlFilters}(\mathcal{E}_{W_{i+1}});\)
11. \(\mathcal{W}_{i+1} \leftarrow \text{nexTets}(\mathcal{E}_{W_{i+1}}, \mathcal{W}_i);\)
12. \(\mathcal{E}_{W_i} \leftarrow \mathcal{E}_{W_{i+1}};\)
13. \(\mathcal{W}_i \leftarrow \mathcal{W}_{i+1};\)
14. if \(\mathcal{W}_i = \emptyset\) then
15. \(\quad \text{done} \leftarrow \text{true};\)
16. end
17. if reachSingularityWrong(\(\mathcal{W}_i\)) then
18. \(\quad \text{exit};\)
19. end
20. end

Figure 7: Edge cutting for the first tetrahedron \(t_0\) in Algorithm 1. \(F_c\) is shown at the center of \(t_0\), \(d_0\) is colored in red and \(n_0\), the closest vector of \(d_0\), is colored in green.

In the example on the right, the first tetrahedron \(t_0\) is shown in dark transparency with its cutting plane, and the tetrahedra of \(\mathcal{W}_0\) are shown in light grey. We then enter an edge-based process where surface \(S_c\) is propagated by successive waves (from line 5 to 19 in Algorithm 1).

First we define \(\mathcal{E}_{W_{i+1}}\) the output intersection triplet of \(S_c\) in the tetrahedra of \(\mathcal{W}_i\). By analogy, \(\mathcal{E}_{W_i}\) gathers all the input intersection points for the wave tetrahedra adjacent to an edge of \(\mathcal{E}_0\) and not already traversed. These tetrahedra are the next wave to perform (denoted \(\mathcal{W}_i\) at line 2).

And we note \(\mathcal{E}_{W_{i+1}}\) all the edges that were used to generated this wave (line 3).
Then we cut each tetrahedron \( t_k \in \mathcal{W}_i \). The cutting method is different from the one used for \( t_0 \). While a local plane was created at the center of \( t_0 \), now each tetrahedron \( t_k \) has already been cut along edges of \( \mathcal{E}_{\mathcal{W}_i} \). At least one edge of \( t_k \) is so already intersected by \( \mathcal{S}_c \). The propagation scheme we use is based on face propagation. It is applied in function \text{cutTet} used at line 8 of Algorithm 1 and is detailed in Algorithm 2.

**Algorithm 2: cutTet**

Data: \((t_k, \mathcal{E}_W)\)

Result: \(\mathcal{E}_{t_k}\)

1. \( \mathcal{E}_{t_k} \leftarrow \emptyset; \)
2. \textbf{foreach} edge \( e \in t_k \) \textbf{do}
   3. \textbf{if} \( e \in \mathcal{E}_{\mathcal{W}} \) \textbf{then}
      4. \textbf{forall} face \( f \in t_k \) \& \( e \in f \) \textbf{do}
         5. \( \mathcal{E}_{t_k} \leftarrow \mathcal{E}_{t_k} \cup \text{fieldApprox}(f); \)
   6. \textbf{end}
3. \textbf{end}
4. \textbf{return} \(\mathcal{E}_{t_k}\)

Propagation is done along each face of the current tetrahedron \( t_k \) starting from edges that belong to \( \mathcal{E}_{\mathcal{W}_i} \) (see Figure 8). Note that a face \( f \) of a tetrahedron \( t_k \) could have been already treated for the tetrahedron sharing \( f \) with \( t_k \), whether this tetrahedron belongs to the current wave or to a previous one. We start the cutting algorithm in \( t_k \) with at least one intersected edge \( e \) knowing that \( e \in \mathcal{E}_{\mathcal{W}_i} \) with \((e, \mathbf{n}, \beta)\). For each face \( f \in t_k \) that is incident to \( e \) and not already treated, we compute an approximation of the cut in the frame field by using a 4th-order Runge-Kutta method (line 5 of Algorithm 2). It gives us an output intersection point in \( f \). Let us note that if two edges of a face \( f \) are cut while \( f \) has not yet been traversed, we add \( f \) to the set of treated faces in order to avoid inconsistencies\(^4\).

As we cut geometrically through faces, it may happen that the output intersection point along an edge is located on a node \((\beta = 0 \text{ or } 1)\) of \( \mathcal{T} \). In practice, we avoid such an intersection by randomly moving the intersected nodes. In our implementation, we store all the nodes that are intersected during a wave \( \mathcal{W}_i \) in a set \( \mathcal{N}_i \) and we move all of them simultaneously with small random perturbations. After that, we compute the cut propagation in the tetrahedra of \( \mathcal{W}_i \) that are adjacent to a node of \( \mathcal{N}_i \). This process is repeated until having \( \mathcal{N}_i = \emptyset \). In our different experiments, this set was always empty after a few number of iterations. Note that in our implementation, a node is considered intersected if the intersection point is located at a distance less or equal to \( l = \alpha \cdot \text{length}(e_{\text{min}}) \) where \( e_{\text{min}} \) is the shortest edge of \( \mathcal{T} \) and \( \alpha = 0.1 \). Figure 6 illustrates different stages of the dual sheet propagation for a regular grid-like frame field.

**Singularity line traversal.** During the wave propagation, we can reach some singular tetrahedral cells (line 17 of Algorithm 1). If the tetrahedral cell contains a singularity point, we stop the wave propagation.

Otherwise, it means that the tetrahedron is traversed by a singularity line. In this case, we compare the normal vector to the dual sheet with the singularity line direction: if they are almost aligned (dot product evaluation), then the dual sheet can traverse the singularity line, else we stop the wave propagation. In our implementation, the normal to the dual sheet and the singularity line direction are considered aligned when their angle is less than \( \pi/4 \).

### 3.2 Control filters

Our dual sheet creation process consists in creating successive waves. Each time we go through a tetrahedron, numerical approximations are done and small geometrical errors may appear. While they have a very little impact locally, their accumulation can lead to unexpected results. In particular the dual sheet can break and split (see Figure 9). In order to prevent such a situation to occur, we apply some filters to remove some data from \( \mathcal{E}_{\mathcal{W}_i} \) at the end of each wave (line 10 of Algorithm 1).

We have two filters in our algorithm to control the wave propagation. The first one is based on the topology of \( \mathcal{T} \); the second one considers the dual sheet geometry. During the dual sheet creation, we build successive waves. Each of these has a number, and all

\(^4\) A third intersection must not be added, it would introduce unexpected configurations that could be difficult to handle.
Figure 9: Impact of the geometric filter on the dual sheet propagation with the tetrahedra of the dual sheet on the left column and the created intersection points on the right. First row shows the result without applying the filter. Second and third rows respectively correspond to $\lambda_g = 0$ and $\lambda_g = \sqrt{2}$. The tetrahedral cells that belong to a wave are labeled with this wave number. When we create the set $W_{i+1}$, the topological filter removes from $W_{i+1}$ all the tetrahedra that would share an edge, a face or a node with a tetrahedron belonging to $W_j$, with $j \in [0, i + \lambda_t - 1]$ and $\lambda \in \mathbb{N}^*$. In our implementation, we fixed $\lambda_t = 2$.

The second filter controls the geometric propagation of each point in $E_{W_{i+1}}$. Let $p_i$ be such a point (see Figure 10). We store for this point the two previous points used to generate it: the point $p_{i-1}$ that belongs to the same tetrahedron of $M_i$ as $p_i$, and the point $p_{i-2}$ that belongs to a tetrahedron of the previous wave $M_{i-1}$. We then check the deviation between the previous direction $d_{i-1}$ and the current direction $d_i$. The edge $e$ is kept in $E_{W_{i+1}}$ if $\frac{d_{i-1} \cdot d_i}{|d_{i-1}| \cdot |d_i|} > \lambda_g$, with $\lambda_g \in [0, 1]$. Having $\lambda_g > 0$ prevents the sheet from turning 90 degrees and eventually geometrically turning back in the domain. We can notice that this filter does not take the frame field $F$ into account. This control and the topological control are mandatory near singularity lines where high curvature in the frame field induces much more difficulties to preserve the dual surface topology.

4. DUAL BLOCKING PROPERTIES

Applying the algorithm described in Section 3 allows us to partition the domain into blocks. But not all partition can produce a hexahedral block structure. This is why we have to check a minimal set of properties about the obtained block structure.

Figure 10: Geometric filtering based on predecessors.

Figure 11: Successive dual curve insertions to build a quad block structure.

4.1 Dual structure properties

In order to explain our approach, let us consider the 2D example of Figure 11. In 2D, dual sheets are dual curves and the dual of a quad block structure can be seen as a simple arrangement of curves. In (a), we consider that the user has only created one single dual curve. Obviously, this configuration does not correspond to a valid dual structure since at least one dual
node, i.e. one primal face, must exist. In (b), a second curve is so added and the corresponding primal block structure is given in (c). It is reduced to one quadrilateral block, which is a valid block structure. But it does not fit the frame field prescription. By construction, the input frame field has a singular triangle in the domain (shown in red on Fig. 11) and the usually expected block structure is the one given in (e). In order to get this primal structure, one dual curve is missing: this line is added in (d) in such a way that the singular red triangle is enclosed by dual curves into an inner-surface dual zone. In other words, a boundary dual zone should not contain a frame singularity.

In a similar way, a set of validity rules can be defined in 3D. In order to define this set of rules, let us first introduce the notion of classification. Let \( \mathcal{T} \) be a tetrahedral mesh of a domain \( \Omega \), where the boundary \( \partial \Omega \) is made of a set \( \partial \Omega_2 \) of surfaces, a set \( \partial \Omega_1 \) of curves and a set \( \partial \Omega_0 \) of vertices. For the sake of simplicity, we consider that \( \Omega \) is a single volume and not an assembly of volumes. Considering such a decomposition of \( \Omega \) and a set of dual sheets that split \( \Omega \) into dual zones \( Z_\Omega \), the following criteria must be checked:

1. At least one block exists, i.e. at least three dual sheets intersect in one single dual node;
2. A boundary dual zone can not contain more than one geometric point of \( \partial \Omega_0 \) or one boundary frame singularity point;
3. A boundary dual zone can not contain one boundary frame singularity point and being along a curve.
4. An in-volume dual zone can not contain more than one frame singularity point;
5. An in-volume dual zone that does not contain a frame singularity point can be traversed by one single frame singularity line at most.

First criterion is obvious and avoid peculiar cases where \( \partial \Omega_0 \) is empty (for totally smooth boundary). It is fulfilled by the example shown in Figure 12, where dual zones A, B, C and D are created. As each dual zone corresponds to a primal block node, we meet an issue with dual zone C that contains geometric vertices 1, 2 and 3. We must split this dual zone so has to have each vertex in a different zone. It is the purpose of criterion 2 (see Figure 13). Criterion 3 prevents the situation depicted on Figure 11-b to arise: considering the dual zone containing the red triangle (and so a boundary singularity point), this zone must not be bounded by a geometric curve. Criteria 4 and 5 are relative to in-volume dual zones. In our approach, a dual zone corresponds to a primal node, while a singularity point also corresponds to an expected primal node if you build the blocks from singularity points and lines [8]. If a dual zone would contain two singularity point, we would get an inconsistency. It is the purpose of criterion 4. Criterion 5 is similar. A singularity line should correspond to an edge of the primal block structure. If a dual block \( B \) contains two singularity lines \( l_1 \) and \( l_2 \) but no singularity point, it means that \( l_1 \) and \( l_2 \) do not intersect and it induces that the primal node relative to \( B \) should be on two non-intersecting edges. That is not possible and it is the purpose of criterion 5.
4.2 Validity blocking algorithm

Starting from the dual sheet extraction processed beforehand on tetrahedral mesh \( T \) we want to check the validity of the block decomposition. For each tetrahedron \( t \in T_3 \), we know if it is traversed by 0, 1, 2 or 3 dual sheets. Let \( f_{sh} : T_3 \mapsto [0; 3] \) be the function giving this information. The algorithm we apply is the one provided in Algorithm 3. It gives as a result whether the induced block decomposition is valid or not and two functions \( f_{bl} \) and \( f_c \). Function \( f_{bl} : T_3 \mapsto \text{IN} \) indicates for each 3-cell of \( T \) the block number it belongs to (value 0 means no block). Function \( f_c : \text{IN} \mapsto \text{IN} \times \text{IN}^* \) gives for each block \( b_i \) the dimension and the id of the lowest geometric entity a node of \( b_i \) is classified on.

Algorithm 3: Validity blocking algorithm.

<table>
<thead>
<tr>
<th>Data:</th>
<th>( T, f_{sh} : T_3 \mapsto [0; 3] )</th>
</tr>
</thead>
<tbody>
<tr>
<td>Result:</td>
<td>( f_{bl} : T_3 \mapsto \text{IN}, f_c : \text{IN} \mapsto \text{IN} \times \text{IN}^* )</td>
</tr>
</tbody>
</table>

1. \( bm \leftarrow \text{initBooleanMark}(T_3) \);
2. for \( t \in T_3 \) do
   3. if isMarked\((t, bm)\) and isSingular\((t)\) then
      4. \( B_t \leftarrow \text{colorAndMarkBlock}(t, f_{bl}, bm); \)
      5. \( \{d_{\text{min}}, \{i_{\text{min}}\}\} \leftarrow \text{getMinClassification}(B_t); \)
      6. \( \{n_{sp}, n_{sl}\} \leftarrow \text{getSingularityData}(B_t); \)
      7. if !checkValidity\((d_{\text{min}}, \{i_{\text{min}}\}, n_{sp}, n_{sl})\) then
         8. return false;
   end
3. end
4. releaseBooleanMark\((bm, T_3)\);
5. return true;

Algorithm 3 relies on a traversal of all the tetrahedra of \( T_3 \). Let \( t \) be a non-traversed and non singular tetrahedron, we color all the tetrahedra that belong to the same block as \( t \) (line 4). This set of tetrahedra, denoted \( B_t \), is built by a breadth-first traversal starting from \( t \) and using edge connectivity. During the traversal, a tetrahedron \( t' \) is added to \( B_t \) if \( f_{sh}(t') = 0 \). Then we check the minimal classification \( (d_{\text{min}}, \{i_{\text{min}}\}) \) of the nodes of tetrahedra in \( B_t \) (line 5). Value \( d_{\text{min}} \) gives the minimal dimension (0 for vertex, 1 for curve for instance); the set \( \{i_{\text{min}}\} \) contains the id of the \( d_{\text{min}} \)-dimensional geometric entity. For instance, getting \( (0, \{1, 4, 2\}) \) would mean that some nodes of \( B_t \) are classified on geometric vertices 1, 4 and 2. At line 6, we get the number of singularity points \( n_{sp} \) and the number of singularity lines \( n_{sl} \) that contain tetrahedra of \( B_t \). With data gathered at line 5 and 6, we can check the validity of the rules previously given. For instance: \( n_{sp} \) must be less or equal to 1; and if \( d_{\text{min}} = 0 \), \( \{i_{\text{min}}\} \) must be equal to 1.

5. BLOCK EXTRACTION

Once dual zones are validated and classified through function \( f_c \), we have to create the primal nodes and the primal hexahedral blocks.

5.1 Primal node creation

Every tetrahedron \( t \in T_3 \) is in a dual zone \( (f_{sh}(t) \neq 0) \) or in a dual sheet \( (f_{sh}(t) = 0) \). By construction, a dual zone is discretized as a set of tetrahedra bounded by either dual sheets or domain boundary. From a topological point of view, a dual zone corresponds to a primal node in the block structure. In order to create this node, we use the classification data stored in \( f_c \). Let's consider the dual zone \( i \), and the corresponding primal node \( n_i \), then \( f_c(i) \) returns the dimension and id of the geometric entity to classify \( n_i \). If the dimension is equal to 0, \( n_i \) is located on the corresponding geometric point. Otherwise, we initialize \( n_i \) location at the center of mass of dual zone \( i \) and if the dimension is equal to 1, respectively 2, we project \( n_i \) onto the corresponding geometric curve, respectively the corresponding geometric surface.

5.2 Dual zone labeling

We build primal blocks from the dual structure directly. More precisely, we identify each dual zone by a label that is a series of dual sheets numbers and we compare this series with the numbers of the dual curves that intersect each other at \( n \). Dual zones are labeled by a \( p \)-uple of numbers, where each number corresponds to a bounding dual curve.

![Figure 15: Dual zone numbering for building primal blocks. The label of a dual node \( n \) is a couple of 2 dual sheet numbers (in red), which are the numbers of the dual curves that intersect each other at \( n \). Dual zones are labeled by a \( p \)-uple of numbers, where each number corresponds to a bounding dual curve.](image)
with two numbers\(^5\). For instance, the dual node intersected by dual sheet 1 and 2 has label 1 – 2 (the dual node on the bottom left on Figure 15). Dual zones are labeled with the numbers of the dual sheets that surround them. As a consequence the label size of a dual zone is not fixed and this label is not necessary unique. For instance, the symmetry of the domain exhibited on Figure 16 induces that dual curve 1 intersects dual curve 2 twice. Two dual nodes have so the label 1 – 2 and all the zone labels are carried by two dual zones. This labeling strategy extends to 3D where dual nodes are labeled with a triplet of dual sheets numbers\(^6\).

![Figure 16](image1)

**Figure 16:** As two dual lines can intersect each other more that once, dual node and dual zone labeling is not unique.

Our method is quite similar to [5], where they label the dual zones, in a global way, according to their relative locations to all dual sheets, that are represented by implicit surfaces. They declare a dual zone as being on one side of a dual sheet (flag +) or on the other side (flag −). If you have \(n\) dual sheets, each zone is then labeled by a \(n\)-size vector of flags \((+,-)\). This strategy is not adapted to our algorithm since it does not allow us to handle self-intersecting dual sheets and it doesn’t fit our inserting dual sheets for volumes that are not homeomorphic to a sphere. In the example of Figure 16, dual lines 3 and 4 should be merged into a single line to apply the strategy given [5] and get the right labeling for us. With our local strategy where we label each dual zone with the labels of the dual sheets it is enclosed in, we do not get such an issue.

### 5.3 Primal block creation

The label assignment previously done partially carries the topology of the dual structure and so of the primal hexahedral blocks. In order to create primal block, we adopt a two-stages procedure where: (1) First, we build hex corners for every node \(a\), which are 3-tuples of primal nodes \(C^*_a = (c_1, c_2, c_3)\) corresponding to a future block corner. In other terms, it means that \([a, c_1], [a, c_2]\) and \([a, c_3]\) will be three edges of a primal hexahedron; (2) Then, we build hexahedral blocks by combinatorial operations on hex corners.

**Creation of hex corners.** Each 3D dual node corresponds to a primal block. It is surrounded by 8 dual zones, each of them corresponding to a primal node that has been created at Section 5.1. We label each 3D dual node \(n\), respectively a 2D one, with 3 numbers, resp. 2. These numbers correspond to the dual sheets, resp. line, that intersect at node \(n\). Let \(l_n\) be the label of \(n\). In order to find out the dual zones surrounding \(n\), we traverse all the dual zone labels and we keep those having \(l_n\) as a sub-series of their own label. Let \(Z_n\) be the set of dual zones surrounding the dual node \(n\). If the dual node labeling is unique, i.e every dual node has a different label, then we have the standard configuration where \(|Z_n| = 8\) in 3D, resp. 4 in 2D, for all dual nodes.

But the dual node labeling can be not unique as soon as two dual sheets in 3D, resp. dual lines in 2D, intersect each other twice or more. It is the case on Figure 16 where two nodes have the label 1 – 2 because lines 1 and 2 intersect twice. Both of those nodes gather all the dual zones of this simple model into their respective sets \(Z_n\). And we get \(|Z_n| > 8\) in 3D, respectively \(|Z_n| > 4\) in 2D. In order to reduce \(Z_n\) size to 8 in 3D, resp. 4 in 2D, we compute a distance between each zone of \(Z_n\) and \(n\) and we keep the 8 in 3D, resp. 4 in 2D, closest ones. This distance is computed as follows. The dual node \(n\) is located into a simplex \(s_n\) of \(\mathcal{T}\). We pick the center of mass of \(s_n\) as an approximated location for \(n\). Let \(p_n\) be this location. Each dual zone being a set of simplices\(^7\) \(S\) in \(\mathcal{T}\), we compute the Euclidean distance between \(p_n\) and each center of mass of an element of \(S\) and we keep the smallest distance.

![Figure 17](image2)

**Figure 17:** Final blocking for the models given in Figures 15 and 16.

At this stage, every primal node has a final geometric location and every primal block has the list of the pri-

\(^5\)Note that it can be twice the same number when a dual curve intersects itself.

\(^6\)The same number can occur twice in case of self-intersecting dual sheets.

\(^7\)triangles in 2D and tetrahedra in 3D.
Figure 18: 2D example of dual sheet traversal in order to build corner data. Here, blue triangles belong to 2 dual lines, while red ones are in a single dual line and in contact with dual zone A. Those red triangles allow our algorithm to connect dual zone A to dual zones B and C to form a quad block corner.

Creation of primal blocks. Now that hex corner data is built, we build faces of blocks. On Figure 19-c, we start from node A with hex corner $C_A = (C, D, E)$. Considering the node A, two of the neighbor nodes are selected in $C_A$, which are C and E here (see Figure 19-c). We can then obtain the first face by getting the node of $C_C \cup C_E - A$, which is F (see Figure 19-d). The first face is so $AECF$. We build two other faces adjacent to A by selecting D and E in $C_A$, then C and D. Those faces are respectively $AEHD$ and $ACGD$ (see Figure 19-e). The final step is to create the last faces with the remaining node B. Such a process applied in 2D gives blocks of Figure 17 for the examples of Figures 15 and 16.

6. EXPERIMENTAL RESULTS

We applied our approach on several CAD models. They were imported into the gmsh software [25] using the step file format and gmsh was used to generate initial tetrahedral meshes. Discrete frame fields were then computed on each tetrahedral mesh by a method similar to the one described in [10]. Each tetrahedral mesh $T$ enriched by a frame field defined at the nodes of $T$ is then taken as the input of our software. The parameter $\lambda_g$ used in the dual sheet creation process is fixed to 0.2 for all the results shown in this section. Some results we obtained are shown on Figures 20, 23, 27 and 28. For every model, we show the tetrahedral representation of dual sheets and dual zones and the final primal hexahedral blocks. Our pipeline of operations - dual sheet creation, block property checking, primal block generation - gives the expected results for those models when the input mesh is refined enough and the frame field is valid. Some statistics about the generation of all the presented results are given in Table 21.

6.1 Impact of the mesh resolution

Two steps are impacted by the mesh resolution: the dual sheet creation and the block extraction. Building a single dual sheet requires to propagate a dual surface into the tetrahedral mesh starting from a single point and direction. This propagation being done numerically by linearly integrating the field along each
Figure 20: Dual structure (left), dual zones (middle) and final hexahedral blocks (right) for several models.

| Model | $|T|$ | Sheets | Dual Zones | $|B|$ | avg | min | max | validity | blocks |
|-------|-----|--------|------------|-----|-----|-----|-----|---------|--------|
| B7    | 82879 | 4      | 15          | 4   | 0.3325 | 0.3  | 0.41 | 0.21    | 0.88   |
| B8    | 74112 | 7      | 33          | 10  | 0.2816 | 0.24 | 0.31 | 0.21    | 1.58   |
| B10   | 355048 | 10     | 40          | 12  | 1.8533 | 1.41 | 2.31 | 1       | 3.81   |
| B28   | 362531 | 24     | 136         | 49  | 1.3209 | 0.45 | 7    | 0.95    | 6.63   |
| B31   | 349598 | 15     | 96          | 37  | 1.7331 | 0.53 | 3.79 | 1.19    | 7.52   |
| B40   | 57984  | 5      | 16          | 3   | 0.2280 | 0.12 | 0.38 | 0.15    | 0.38   |
| B45   | 164904 | 7      | 24          | 6   | 0.6014 | 0.32 | 1.28 | 0.47    | 1.53   |
| S3    | 932672 | 19     | 184         | 97  | 7.3423 | 3.1  | 21.03 | 2.32    | 19.78  |
| S7    | 134494 | 13     | 96          | 51  | 0.5727 | 0.28 | 1.56 | 0.41    | 3.91   |
| S34   | 122399 | 10     | 51          | 17  | 0.6144 | 0.11 | 1.46 | 0.31    | 1.73   |
| S35   | 99829  | 9      | 50          | 16  | 0.3777 | 0.14 | 0.9  | 0.27    | 1.41   |
| S37   | 595953 | 6      | 39          | 14  | 4.5050 | 1.78 | 7.28 | 2.11    | 10.64  |
| S38   | 200245 | 18     | 112         | 44  | 0.70222 | 0.16 | 3.89 | 0.55    | 3.78   |
| S40   | 414542 | 23     | 164         | 74  | 2.1820 | 0.48 | 9.01 | 1.24    | 7.26   |

Figure 21: Statistical results for our method. $|T|$ is the number of tetrahedron in the mesh, $|Sheet|$ the number of dual sheets created, $|Dual Zones|$ the number of dual zone generated and $|B|$ the number of blocks extracted. $\text{avg}$, $\text{min}$ and $\text{max}$ are respectively the average, minimal and maximal time for one sheet creation. $\text{validity}$ is the time of our dual validation algorithm and $\text{blocks}$ the time for the blocks extraction algorithm. All times are given in seconds. Experiments were made on a standard desktop computer.
traversed tetrahedron, their size has an obvious impact on the result. For instance on Figure 22, the "same" dual sheet is created with two mesh resolutions. In (a) with the coarser resolution the sheet creation process stops when it hits a singularity line. In (b) with a refined resolution, the gap between singularity lines and the boundary is larger letting the dual sheet pass.

![Figure 22](image)

**Figure 22:** Extraction of a "same" dual sheet in a tetrahedral mesh of a geometric domain with two resolution levels: coarser in (a) than in (b).

Figure 23 illustrates another impact of the mesh resolution on our algorithm. Contrary to the examples of Figure 20, in both models of Figure 23, we have introduced boundary dual sheets by following both the frame field and the domain boundary. It is the case of the red dual sheets in the left column. A boundary dual sheet is created when we pick a tetrahedron along the domain boundary and we select as a sheet direction the normal to the surface. In this case, the dual sheet extraction algorithm presented in Section 3.1 may fail without adding this control. Indeed if the mesh resolution is too low, it may end up by hitting a singularity line that is too close to the boundary.

As our approach consists in directly creating 3D dual sheets, we can obtain 3D dual sheets bounded by several loops that have different topologies. Two examples are given on Figure 25. For both of them, we don’t know how the 3D dual sheets would be recovered from the exhibited loops as done in [5, 6].

### 6.2 Wrong frame field configurations

The frame field is an input of our algorithm. As a consequence, if the frame field does not fit the requirement of hexahedral blocking, our algorithm fails to generate a block structure. We currently distinguish two main issues that limit the application of our algorithm: 3-5 singularity lines and "ski-ramp" configurations. The 3-5 singularity lines are known characteristics of frame fields that do not correspond to hexahedral block structure [26, 27]. Such a line connects a 3-valent boundary singular point $A$ to a 5-valent singular point $B$. It means that this line should correspond to a series of edges of the primal blocks such as they would be adjacent to 3 hexahedra at $A$ and to 5 hexahedra at $C$. Such a transition is not possible in a full-hex mesh without adding extra singularities. If you restrict the incident edges of a node in a hexahedral mesh to be 3, 4 or 5-valent, authors of [27] enumerate 11 topologically different interior node types. Among those configurations, there is no configuration with only one 3-valent edge and one 5-valent edge (others being regular 4-valent).

As a consequence, our approach only allow the user to draw dual sheets far away from such a line (see Figure 24-left column), leading to an incomplete block solution. Looking at the dual sheet structure (on middle-left), we could expect to have more blocks generated (on the bottom-left). If we relax some validity rules, we can obtain a valid block structure that snaps the singularity line along the boundary surface.

The "ski-ramp" configuration corresponds to a part of geometrical models that where a surface is pinched forming a very narrow angle. In such a situation, the frame field is totally regular, i.e. has no singularity line. As a consequence and all the dual lines we could build coming from the left side of the ski jumps follow the parameterization direction (blue lines) and end up in the single point on the right. It induces that our meshing technique do not work in such a situation since two lines end up at on the same geometric curve (see dual structure on the top of Figure 26).

### 7. CONCLUSION

We have proposed an approach to build block hexahedral meshes in an indirect interactive manner. Starting from a 3D frame field, the user creates 3D dual sheets and the proposed algorithm build a dual structure that is eventually converted into hexahedral blocks. This process is iterative and guided by validity rules the dual structure must respect. It has been validated on several CAD models whose complexity is similar to the ones presented in Section 6. It fundamentally differs from recent interactive attempt to generate hexahedral blocks [5, 6] in the fact that we directly handle 3D dual sheets. This difference induces a totally different pipeline of algorithms and concepts: dual zones, validity rules for the dual structure, primal blocks creations.
Figure 23: Dual structure (left), dual zones (middle) and final hexahedral blocks (right) for several models.

Figure 24: Because of a 3-5 singularity line (top-left), dual sheets are not generated near the singularity line (middle-left) and the block structure is incomplete (bottom-left). By relaxing our validity rules for this type of line, we can obtain a block structure that does not strictly follow the frame field topology (right column).

Figure 25: Examples of 3D dual sheets bounded by several boundary loops.
In order to go further, we will improve our approach following three main directions. First, our dual sheet creation process requires a quite refined mesh. When two singularity lines are separated by only a few number of tetrahedra of $T$, or when a singularity line is too close to the domain boundary, it may happen that the user is unable to insert a dual sheet in this area. We have provided a partial answer by introducing a boundary dual sheet insertion process, but we need a more global solution. Mesh adaption techniques on $T$ with frame field interpolation seems reachable and would give the ability to adapt the mesh when problematic situations occur. Another option is to make the dual sheet creation algorithm more precise on coarse meshes.

The second direction to go through is relative to intrinsic frame field limitations. A frame field input may not corresponds to a valid hexahedral block structure. It happens at least in two situations: (1) very sharp regions with small angles (see Figure 26); (2) when a singularity line connects a 3-type boundary singularity point to a 5-type one (see Figure 24). For the former case, we intend to add 3D interactors to "cut" these parts and replace them by an appropriate dual zone directly. For the latter case, 3D interactors coupled with a frame field modification algorithm should allow us to get a valid dual structure.

The last direction is about user-guidance and interactivity. We intend to provide support to the user when the block structure is detected as being invalid. We could also give to the user some suggestions of regions where to a tetrahedron should be picked for creating a dual sheet. Our current implementation can also be drastically improved to get timings compatible with a fluent interactive process (see Table 21 for our current results).

**References**


Figure 27: Dual structure (left), dual zones (middle) and final hexahedral blocks (right) for several models.


Figure 28: Dual structure (left), dual zones (middle) and final hexahedral blocks (right) for several models.