Available online at www.sciencedirect.com

SciVerse ScienceDirect

Procedia Engineering 00 (2017) 000–000

SAND2017-7060C

Procedia Engineering

www.elsevier.com/locate/procedia

Draft - 26th International Meshing Roundtable

Volume preservation improvement for interface reconstruction hexahedral methods

Nicolas Le Goff[a,*], Franck Ledoux[a], Steven Owen[b]

[a] *CEA, DAM, DIF, F-91297, Arpajon, France*
[b] *Sandia National Laboratories[1], Albuquerque, NM, USA*

Abstract

We propose a new post-processing procedure for automatically adjusting node locations of an all-hex mesh to better match the volume of a reference geometry. Hexahedral meshes generated via an overlay grid procedure, where a precise reference geometry representation is unknown or is impractical to use, do not provide for precise volumetric preservation. A discrete volume fraction representation of the reference geometry M^I on an overlay grid is compared with a volume fraction representation of a 3D finite element mesh M^O. This work proposes a procedure that uses the localized discrepancy between M^I and M^O to drive node relocation operations to more accurately match a reference geometry. We demonstrate this procedure on a wide range of hexahedral meshes generated with the Sculpt code and show improved volumetric preservation while still maintaining acceptable mesh quality.

Peer-review under responsibility of the organizing committee of IMR 26.

Keywords: hexahedral mesh ; interface reconstruction ; volume fraction ; volume preservation ; overlay grid

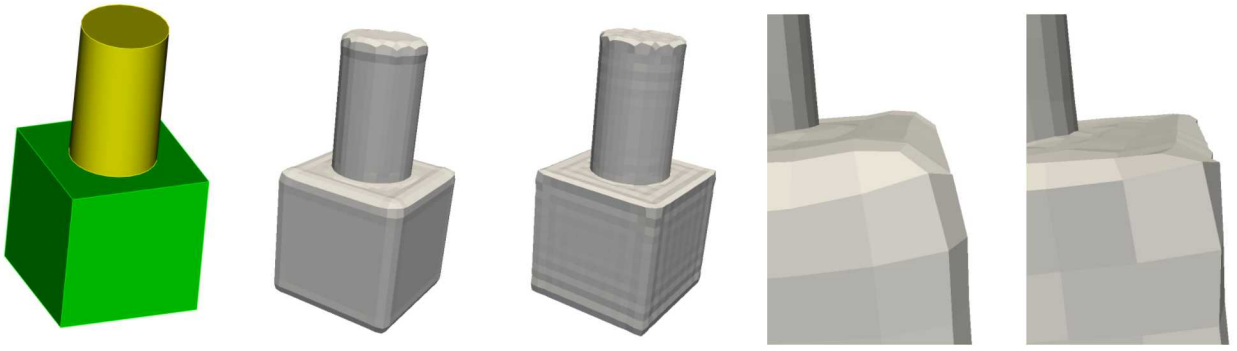
1. Introduction

Overlay grid methods [1–4] developed in recent years have dramatically improved the ability to rapidly and automatically generate hexahedral meshes for complex geometries in massively parallel environments. Overlay grid procedures utilize a *mesh-first* approach to mesh generation where an initial base grid is used to overlay the reference geometry. Procedures to modify the base grid are employed to best capture the geometry to define a conformal all-hex mesh. In contrast, *geometry-first* mesh generation approaches [5–7] rely on user-intensive procedures to first clean and then decompose the geometry to fit blocking or sweeping topologies. Because of the nature of geometry-first approaches, these methods can in most cases very accurately preserve volume of a reference geometry subject only to a user defined mesh resolution, and would therefore benefit little from the proposed work. However, as geometry-first technologies for hex meshing cannot be easily automated or scaled for general use, mesh-first procedures, such as overlay grid are often preferred and consequently the focus of this work.

[1] Sandia National Laboratories is a multimission laboratory managed and operated by National Technology and Engineering Solutions of Sandia, LLC., a wholly owned subsidiary of Honeywell International, Inc., for the U.S. Department of Energys National Nuclear Security Administration under contract DE-NA0003525.

* Corresponding author. Tel.: +33-169-264-000.
E-mail address: nicolas.le-goff@cea.fr

Peer-review under responsibility of the organizing committee of IMR 26.



(a) (b) (c) (d) (e)

Fig. 1. (*a*) CAD model for the brick-cylinder case (*b*) Input mesh M^I generated by Sculpt (*c*) Output mesh M^O based on proposed method (*d*) Close-up of one corner of input mesh M^I (*e*) Close-up of same corner of mesh M^O. Note that proposed method better represents geometric corner.

Some overlay grid procedures, such as Sandia's Sculpt [4] algorithm, use an interface reconstruction procedure that relies on a volume fraction representation of the geometry on a Cartesian or adaptively refined grid. The primal contouring approach described in [4] will adjust nodes on a base grid to conform to an approximation of the reference geometry prior to application of pillowing and smoothing operations. Because of the approximate nature of the interface reconstruction procedure combined with smoothing, the resulting all-hex mesh may not precisely conform to the reference geometry. While in most cases Sculpt meshes have proven accurate in simulation compared to pave and sweep approaches [8], we note one potential deficiency. In some cases where localized densities and material properties demand accurate volume preservation, the interface reconstruction employed by Sculpt and other overlay grid algorithms may not provide sufficient precision.

For our purposes we consider both *explicit* and *implicit* geometry representations with multiple components or materials. Explicit geometry includes B-Rep standards such CAD and STL models while implicit can include 3D image data and volume fractions on a Cartesian grid. Both types of input can be meshed using overlay grid methods. For explicit geometry representations, closest-point projection to B-Rep surfaces may be employed to accurately capture the reference geometry and correctly preserve volume. However, we note that projection operations in overlay grid methods can often create topology cases that cannot be adequately smoothed, resulting in inverted elements. For example, these can include cases where more than one face of a hex lies on a single surface or multiple edges of the same hex lie on the same curve. To correct for these instances, special case topology operations are often employed to locally improve quality [9,10]. These operations, while effective in some cases, can be complex and difficult to employ, and in many cases can result in severely distorted elements.

The proposed volume preservation algorithm concentrates instead on relocating nodes of the mesh to more accurately represent the underlying reference geometry without the need for complex topology operations. The generalized approach we propose provides both for explicit geometry, and for implicit geometry where only the localized volume fraction information is known and where exact closest-point operations are otherwise not practicle.

2. Definitions and notations

In this work we consider only all-hex meshes, however as the proposed solution is not limited to hexahedral elements, we adopt the following definitions. In dimension 3, we call *nodes* the 0-dimensional cells, *edges* the 1-dimensional cells, *faces* the 2-dimensional cells and *cells* the 3-dimensional cells. For 2D examples, faces will refer to 1-dimensional cells.

In our context, meshes are used to discretize a physical space made of several materials. Let $\mathcal{M}$ be the set of materials that disjointly fill a geometric domain Ω and M^I a mesh that discretizes Ω, meaning that each cell of M^I can be filled with one or several materials. A cell containing a single material is said to be *pure*; otherwise it is said to be *mixed*. Let c_j^I be a cell of M^I, the *volume fraction* of a material m in c_j^I is the proportion of the volume of m in c_j^I. We

denote $f_{j,m}$ the volume fraction of material m in c_j^I and we have

$$f_{j,m} = f(j,m) = \frac{V(c_j^I \cap m)}{V(c_j^I)} \qquad \text{and} \qquad \sum_{m \in \mathcal{M}} f_{j,m} = 1, \tag{1}$$

where $c_j^I \cap m$ is the geometric intersection between c_j^I and m.

3. Volume-control algorithm overview

For the purposes of this study, we propose a method for improving volume conservation of an ouput mesh that has been constructed using an all-hex overlay grid method. Such algorithms start from a 3D input mesh M^I where each cell can be pure or mixed. As an output, they will produce an unstructured hexahedral mesh M^O, where each cell is pure, i.e. "filled" by only one material. Although meshes produced in this manner will maintain watertight, smooth and manifold interfaces between materials, they do not precisely control for the overall volume of the mesh. We note that other interface reconstruction techniques, such as volume of fluids[11], do indeed precisely control for volume, however rather than producing pure computational elements, they will often yield mixed elements where local interfaces are defined by discrete planar geometry.

Remark. For purposes of this study, we consider M^I as representing a 3D Cartesian grid, however an unstructured mesh may alternatively be used.

3.1. A brief presentation of the overlay-grid strategy

To illustrate the general behavior of overlay-grid algorithms, we consider the Sculpt algorithm [4], which handles both implicit and explicit geometry representations. Sculpt uses an interface reconstruction procedure that relies on a volume fraction representation of the geometry on a Cartesian or adaptively refined grid. Let us consider Figure 2 to illustrate the basic Sculpt procedure, beginning with a Cartesian grid as the input mesh M^I, shown in Figure 2.a. Provided as input, or computed from a CAD or STL description, volume fractions that satisfy equation (1), serve as the basis for the Sculpt procedure. Figure 2.b shows a representation of a field of gradient vectors that are computed from the scalar volume fraction data using finite differences of neighboring cells and a least squares fit of the localized data. Locations where interfaces will most likely cross the virtual edges connecting cell centers are then computed as illustrated in 2.c. Using the local gradient and edge cross locations, node locations of the base grid are repositioned to approximate the interfaces of the reference geometry as shown in 2.d. Figure 2.e then shows conformal layers of hexes or pillows inserted at the interfaces to provide additional degrees of freedom to allow for improvement using smoothing. Finally, in Figure 2.f, combined Laplacian and optimization-smoothing operations are performed, constraining nodes at interfaces to remain on the approximated surfaces and interior nodes repositioned to optimize mesh quality.

3.2. Discrepancy definitions

For many physical applications, it is mandatory to preserve as best as possible the volume of each material during the creation of M^O from M^I while preserving their locality. In other words, the volume of a material m should be the same in M^O and M^I and it should be at the same geometric location. We define volume conservation for our purposes as minimizing the quantity

$$\Delta V = \sum_{m \in \mathcal{M}} |\Delta V_m| = \sum_{m \in \mathcal{M}} |V_m^O - V_m^I| \tag{2}$$

where V_m^I and V_m^O are respectively the volume of material m in meshes M^I and M^O. We note that minimizing ΔV could be done in a global manner. However to preserve material locality, we follow an approach localized to each cell. This process has two benefits: (1) improves the volume conservation of every material, and (2) better controls interface displacements, thus preserving the geometric locality of each material. We observe that conceptually, the objective of the proposed solution is to project back each cell of M^O onto M^I to compute a *discrepancy* value localized to each cell of M^I.

Let c_j^I be a cell of M^I and m be a material, we note $d_{j,m}$ the discrepancy of c_j^I relatively to material m and we define it as

$$d_{j,m} = d(c_j^I, m) = V(c_j^I \cap M_{|m}^O) - f_{j,m} V(c_j^I) \tag{3}$$

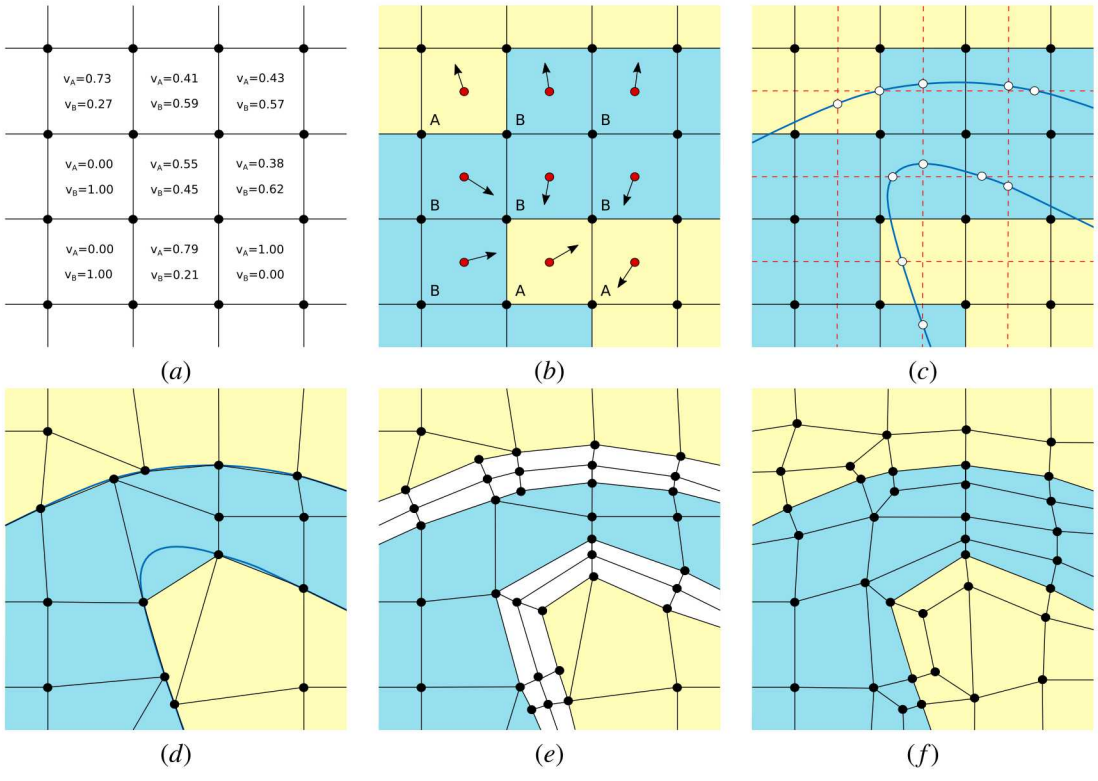


Fig. 2. Sculpt mesh generation process (*a*) Initial volume fraction input on a Cartesian grid. (*b*) Material gradient vectors computed and cells changed to *pure* based on dominant material (*c*) Virtual edge cross locations computed (*d*) Grid nodes moved to approximated interface (*e*) Pillow layers inserted (*f*) Smoothing performed.

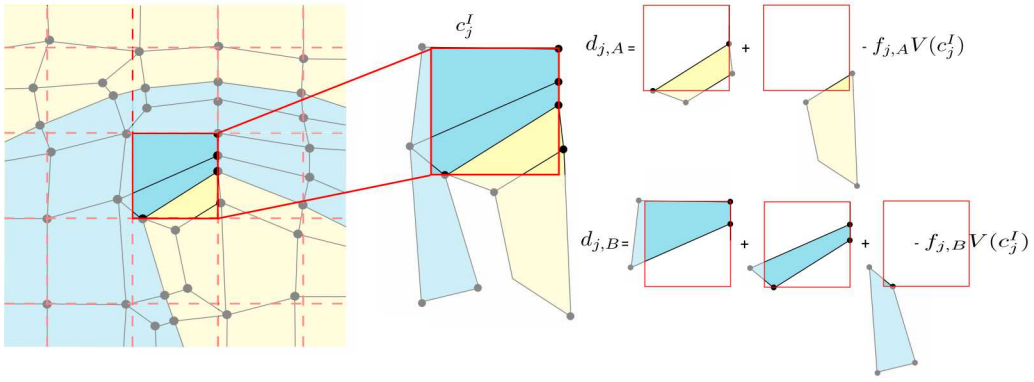


Fig. 3. Considering one input cell c_j^I of the example of Figure 2, we graphically illustrate the discrepancy computation for materials *A* and *B* locally to the cell c_j^I.

where $V(X)$ is the volume of any geometric space X, $M_{|m}^O$ is the output mesh restricted to the pure cells of material m and $c_j^I \cap M_{|m}^O$ is the geometric intersection of c_j^I with the cells of $M_{|m}^O$. We note that for this work, geometric intersections are performed using [12][2]. If $d_{j,m} > 0$, it indicates that locally to c_j^I, we have too much of material m. In contrast, if

[2] The interested reader can directly use the open-source library *portage* [12] based on *R3D* [13].

$d_{j,m} < 0$, it indicates that locally to c_j^I, we do not have enough of material m. The total discrepancy of a cell c_j^I can then be evaluated as

$$d_j = d(c_j^I) = \sum_{m \in \mathcal{M}} |d_{j,m}|, \tag{4}$$

and a global discrepancy can be evaluated as

$$d = d_{tot} = \sum_{c_j^I \in M^I} d_j. \tag{5}$$

Incidentally, replacing $d_{j,m}$ in Equation 4 by its expression from Equation 3 and applying Minkowski's inequality gives :

$$d_j = \sum_{m \in \mathcal{M}} |V(c_j^I \cap M_{|m}^O) - f_{j,m} V(c_j^I)| \leq \sum_{m \in \mathcal{M}} |V(c_j^I \cap M_{|m}^O)| + \sum_{m \in \mathcal{M}} |f_{j,m} V(c_j^I)|, \tag{6}$$

which in turn gives

$$d_j \leq 2V(c_j^I). \tag{7}$$

3.3. Algorithm structure

The following is a brief overview of the proposed volume preservation procedure. The solution is an iterative method where each iteration aims to reduce the total discrepancy. Algorithm (1) and the following points outline the proposed structure of the algorithm:

1. At each iteration, we first improve the quality of M^O by moving some inner nodes (line 3), which are nodes adjacent to only cells of the same material. In practice we use a smart Laplacian [14] or the GETMe agorithm [15] to improve mesh quality.
2. Global discrepancy is computed (line 4) and stored in order to check potential regression during the incoming stage (lines 14 to 17). A regression occurs when the discrepancy does not diminish during two successive iterations.
3. We evaluate the expected target volume of each cell $c_i^O \in M^O$ individually (lines 5 to 7). This process requires intersection of c_i^O with cells of M^I. The full process is described in Section 4.
4. Nodes of M^O are then moved following the algorithm described in Section 5 (line 9). If some movements induce poor mesh quality (computed at line 10), they are withdrawn (lines 11 to 13 for the node update and line 8 for the previous node location storage).
5. Each iteration stage ends with discrepancy evaluation to measure potential regression (line 14). A regression implies canceling the last stage (lines 15 to 17).

4. Target volume of output cells

For a cell $c_j^I \in M^I$, the discrepancy d_j indicates how acurately materials are preserved inside of c_j^I. For a specific material m, the discrepancy $d_{j,m}$ indicates how accurately m is preserved locally to c_j^I. It is this second quantity that we use for computing target volumes for the cells of M^O. Let $c_i^O \in M_{|m}^O$, as c_i^O is a pure cell, it contains a single material m. We compute its target volume as being:

$$T_V(c_i^O) = V(c_i^O) \; - \sum_{c_j^I \in M^I} d_{j,m} \frac{V(c_i^O \cap c_j^I)}{V(M_{|m}^O \cap c_j^I)} = V(c_i^O) \; - \sum_{c_j^I \in \{c^I \in M^I \mid c^I \cap c_i^O \neq \emptyset\}} d_{j,m} \frac{V(c_i^O \cap c_j^I)}{\sum\limits_{c_k^O \in \{c^O \in M_{|m}^O \mid c^O \cap c_j^I \neq \emptyset\}} V(c_k^O \cap c_j^I)} \tag{8}$$

where $\{c^I \in M^I \mid c^I \cap c_i^O \neq \emptyset\}$ are the cells of M^I that intersect c_i^O and $\{c^O \in M_{|m}^O \mid c^O \cap c_j^I \neq \emptyset\}$ are the cells of $M_{|m}^O$ that intersect c_j^I. Figure 4 illustrates equation (8) where we consider the output cell c_i^O. This cell intersects two input cells c_1^I and c_2^I. For each of these cells, we compute a volume contribution, which is one term of the sum in the right member in equation 8. Let us consider cell c_1^I for instance. Discrepancy $d_{1,m}$ indicates the quantity of material m that is under ($d_{1,m} < 0$) or over ($d_{1,m} > 0$) represented in c_1^I.

Algorithm 1: Global structure of the proposed algorithm.

Data: M^I, M^O, maxIter, ϵ
Result: Geometrically modified M^O

```
iter ← 0; regression ← false;
while iter < maxIter & regression == false do
    smoothInside(M^O);
    d_IN ←computeDiscrepancy();
    for c_j^O ∈ M^O do
        T_V[c_j^O] ←computeTargetVolumes(c_j^O, M^I) /* see Section 4 */;
    end
    prev ←storeNodeLocations(M^O);
    moveNodes(M^O, T_V) /* see Section 5 */;
    q ← computeQuality(M^O);
    if q < ε then
        updateSomeNodes(M^O, q, prev);
    end
    regression← (computeDiscrepancy()> d_IN);
    if regression == true then
        updateAllNodes(M^O, prev);
    end
    iter ← iter + 1;
end
```

For example let us consider the first case, which is that $d_{1,m} < 0$. It indicates that we do not have enough material m in c_1^I. Consequently we must inflate the cells of M^O that contain the material m and that intersect c_1^I. These are the cells c_i^O, c_p^O and c_q^O in our example. The *inflate* weight given to each cell c_i^O, c_p^O and c_q^O by c_j^I is proportional to their geometric intersection with c_1^I. For instance, for c_i^O, it is:

$$\frac{V(c_i^O \cap c_1^I)}{V(c_i^O \cap c_1^I) + V(c_p^O \cap c_1^I) + V(c_q^O \cap c_1^I)}. \tag{9}$$

The same computation is also done for the input cell c_2^I. We also observe that as the discrepancy is negative when we do not have enough material in an input cell, the right member of equation (8), equal to $T_V(c_i^O)$, is greater than $V(c_i^O)$.

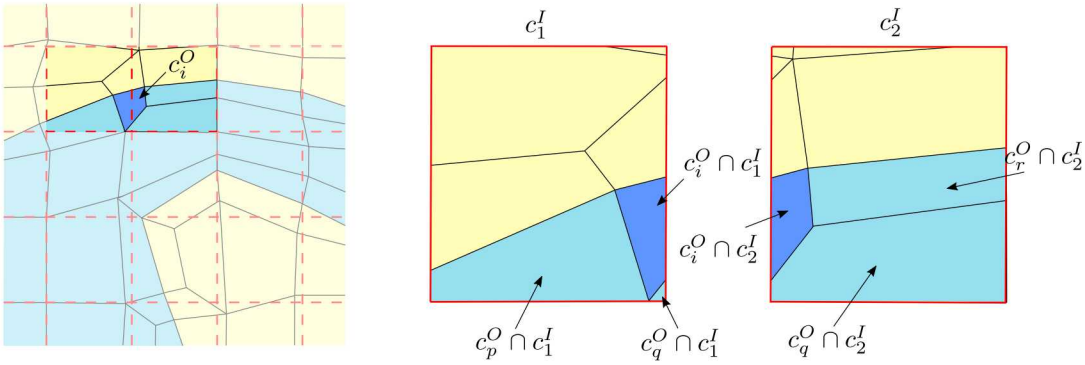


Fig. 4. Illustration of the different geometric quantities used to compute the target volume of the cell c_i^O.

5. Node displacement

Recalling that our objective is to achieve as accurately as possible the volume of input materials. If we consider the output mesh M^O, where each cell is pure, it is necessary to adjust interfaces between materials in order to better preserve material volumes. To accomplish this we can move nodes that are located at the interface between distinct materials. Let n be such a node (see Fig. 5). It is surrounded by pure cells $\{c_1^O, \ldots, c_4^O\}$, which are all assigned to a specific material and have a target volume computed during the previous stage of the algorithm. The procedure used to move nodes is described in Algorithm 2 and depicted on Figure 5. It consists in three main stages:

1. First, we compute an *ideal* deformation for every cell c_i^O having at least one face lying on an interface between distinct materials (lines 1 to 9). Let $\{f_1, \ldots, f_k\}$ be those faces, the deformation consists in defining one translation vector for each face f_i, with $1 \leq i \leq k$.
2. By construction, each interface face f receives two translation vectors $\mathbf{v_1}$ and $\mathbf{v_2}$, one per adjacent cells[3] (except when the void material is not explicitly meshed in M^O, in that case there is only one vector). We assign the average translation vector $\mathbf{t}_f = \frac{\mathbf{v_1}+\mathbf{v_2}}{2}$ to f (lines 10 to 13).
3. Finally, each node n that belongs to the interface between distinct materials is moved according to the translation vectors previously assigned to adjacent interface faces (lines 14 to 21). Let $\{f_1, \ldots, f_k\}$ be those faces, then n is translated along $(\mathbf{t}_{f_1} + \mathbf{t}_{f_2} + \ldots + \mathbf{t}_{f_k})/k$.

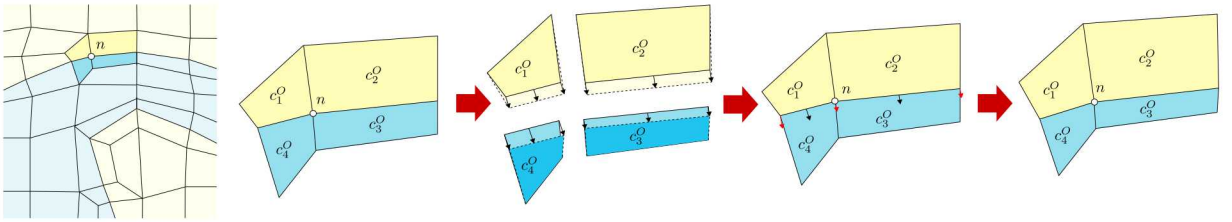


Fig. 5. Illustration of the node displacement procedure. Let n be an interface node and c_1^O, c_2^O, c_3^O and c_4^O its adjacent cells. Ideal shape is computed for each cell c_1^O, c_2^O, c_3^O and c_4^O by moving interface nodes. We then define one translation vector for each interface face, which is averaged at interface nodes to finally move the complete interface.

We drive the computation of the ideal shape of a cell c_i^O along its interface faces. More precisely, we need only move interface nodes during this process using each face's contribution to preserve their normals. To illustrate, let us consider the 2D example of Figure 6 where we list the five configurations we encounter for a quad cell in dimension 2^4. In the first case (see Fig. 6.*a*), only one face is on an interface. As a consequence, only the two red nodes can be moved and they are constrained to follow the face normal (represented in black). In the second case (see Fig. 6.*b*), two faces sharing a node are on an interface. The end nodes can move along their adjacent face normal, while the common node has to follow the sum of the faces normals. The three last configurations are built in the same way (see Fig. 6.*c*, 6.*d* and 6.*e*): a node adjacent to one interface face follows this face normal, while a node adjacent to two interface faces follows the sum of their normal vectors. In other words, the computation of the ideal shape of a cell c_i^O starts by identifying the free nodes $\{n_1, \ldots, n_l\}$ of c_i^O and computing the direction vectors $\{\mathbf{v}_1, \ldots, \mathbf{v}_l\}$ that the nodes will be moved. Once direction vectors are computed, nodes are moved in an iterative way along their respective vectors to fit the target volume $T_V[c_i^O]$ previously calculated. At each iteration, the location of any node n_i, $1 \leq i \leq \ell$, is updated to $n_i = n_i + \kappa \mathbf{v}_i$. The κ term is a dampening term progressively increased until $V(c_i^O)$ reaches $T_V[c_i^O]$. Although the previous discussion illustrates the node displacement procedure in 2D, its extension to 3D can be done in a similar manner.

[3] Which belongs to distinct materials by definition.

[4] Note that the configuration with only one node on an interface cannot occur due to the pillowing process applied during the primal contouring procedure. Were it to happen when using other inputs, we might have to devise another method so as not to ignore the contribution of such a cell.

Algorithm 2: Interface node displacement

Data: M^O, $T_V : M_3 \rightarrow I\!R$
Result: $\mathbf{v} : M_0 \rightarrow I\!R^3$

$t\_map \leftarrow \emptyset$;
for $c^O \in M^O$ **do**
  $V_{c^O} \leftarrow T_V[c^O]$;
  $\{f_1, \ldots, f_l\} \leftarrow$ `getMaterialInterfaceFaces`(c^O);
  $\{\mathbf{t}_{f_1}, \ldots, \mathbf{t}_{f_l}\} \leftarrow$ `computeIdealShape`$(c^O, V_{c^O}, \{f_1, \ldots, f_l\})$;
  for $f \in \{f_1, \ldots, f_l\}$ **do**
    $t\_map[f]$.`add`$(\mathbf{t}_f)$;
  end
end
$f\_map \leftarrow \emptyset$;
for $e \in t\_map$ **do**
  $f\_map[e.$`first`$] = (e.$`second`$[0] + e.$`second`$[1])/2$;
end
for $n \in M^O$ **do**
  $\{f_1, \ldots, f_l\} \leftarrow$ `getMaterialInterfaceFaces`(n);
  $n\_map \leftarrow \emptyset$;
  for $f \in \{f_1, \ldots, f_l\}$ **do**
    n_map.`add`$(f\_map(f))$;
  end
  $\mathbf{v}[n] \leftarrow$ `computeAverageVector`$(n\_map[n])$;
end

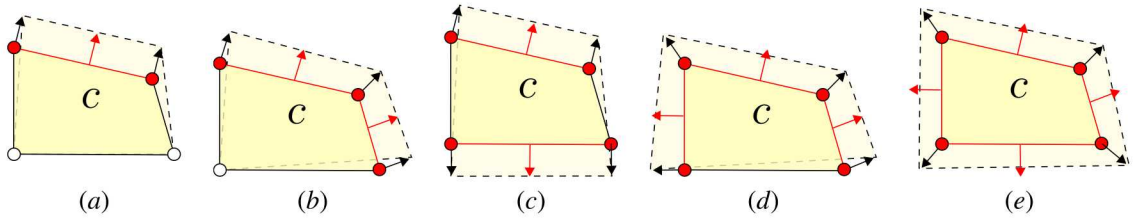


Fig. 6. The five possible configurations to reshape a 2D cell to fit a specified target volume; the remaining other configurations can be obtained by rotation. Interface faces and free nodes are both colored in red. In (*a*), only one face is on the interface leading to only two free nodes that can move along the face normal. In (*b*) and (*c*), two faces are on the interface but in different topological configurations. In (*d*) and (*e*), respectively three and four faces are on the interface.

6. Results

In this section we demonstrate and analyze the results of the proposed method applied to several cases that have been initially meshed using the Sculpt algorithm. The parameters used, as seen in Algorithm 1, use a maximum number of iterations, *maxIter*, of 30 and a quality threshold ϵ that depends on the cases. Since the M^O meshes have different initial minimum scaled Jacobian values, ϵ will be specific to the individual meshes to be improved. For our test cases we used $\epsilon = 0.2$ for the cases illustrated in Table 2 and $\epsilon = \frac{3}{4} scaled\,jacobian_{init}$ for those illustrated in Table 3. In all cases, the κ term, as introduced in Section 5, is equal to $0.1 + 0.9\frac{iter}{maxIter}$.

We first demonstrate the procedure using a simple CAD-based model that consists of the brick and cylinder configuration shown in Figure 7.*a*. Its corresponding Sculpt output M^O_{init} is shown in Figure 7.*b*, while Figures 8.*a* and *b* show the value of the discrepancy ratio per cell of M^I respectively before and after applying our algorithm. We note that the discrepancy decreases after applying our method and is consistent with Figure 9.*a* that shows the distribution

of the discrepancy ratio across the cells. The same is true for the computed target volume adjustment (defined as $T_V(c^O) - V(c^O)$) of the cells of M^O (see Fig. 8.*c* and *d*). Our measures confirm observation from Figures 1.*c* and *d* where the discrepancy is largest around the geometric sharp features both in our input and output M^O mesh. We observe that our solution improves the overall discrepancy, most notably around sharp features.

6.1. Mesh orientation sensitivity

Overlay grid procedures can be particularly sensitive to the orientation of the overlay Cartesian grid with respect to the reference geometry. In this example, we examine the effect of overlay grid orientation on the results of the volume preservation procedure

The following cases use the simple brick-cylinder configuration of the previous example, however apply incremental rotations of 10 degrees to the geometry with respect to the initial overlay Cartesian grid. Tables 1 and 2 illustrate results from varying orientations from 0 to 90 degrees. We observe that our method consistently improves M^O, and note that although discrepancy improvements vary, they are indeed improved in all cases (see Fig. 9 and 10). In particular, the proposed method decreases the total discrepancy as well as the maximum discrepancy ratio per cell.

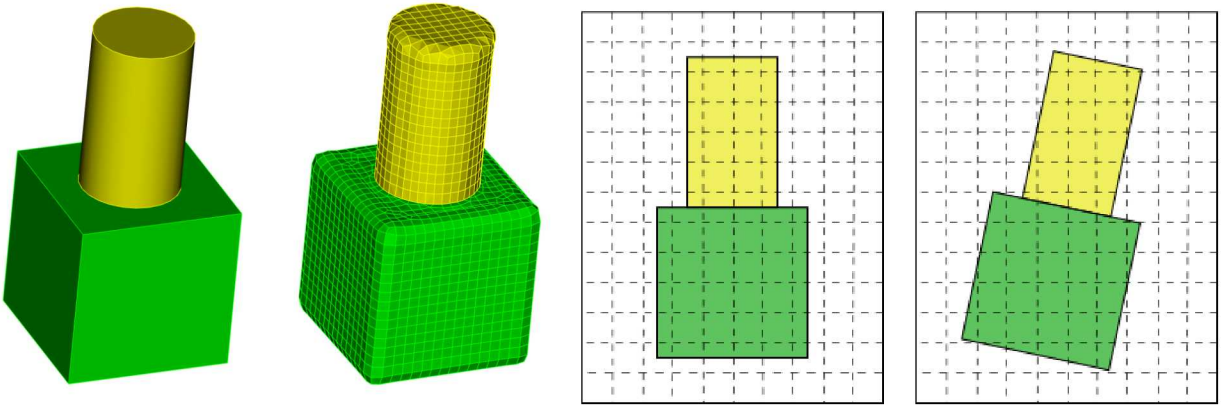


Fig. 7. Brick cylinder example. (*a*) CAD model with two materials (*b*) Sculpt output used as our M^O_{init} mesh (*c*) 2D representation of CAD model with 0 rotation angle (*d*) and 10 degree rotation angle with respect to the overlay Cartesian grid.

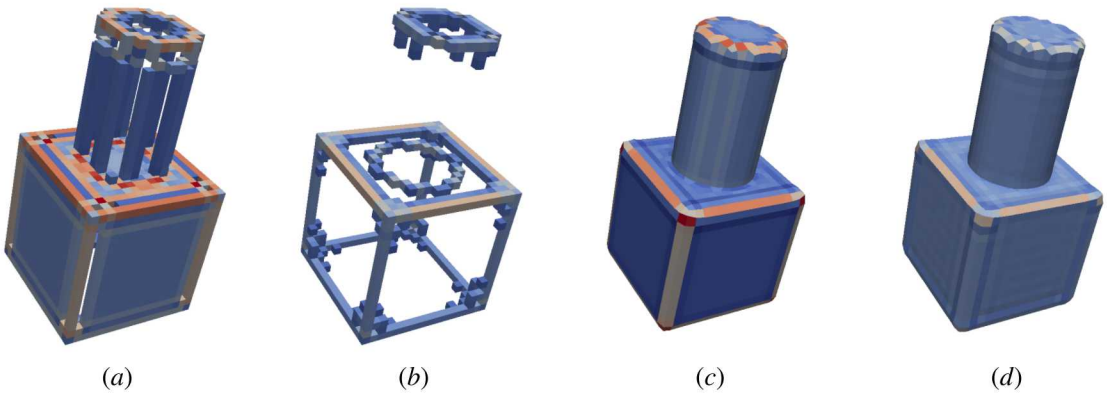


Fig. 8. Results for the brick-cylinder case. (*a* and *b*) initial and final discrepancy (cells $c^I \in M^I$ that have $\frac{d(c^I)}{V(c^I)} < 0.05$ are not represented). We can see that in both the input and output the worst cells are located near the sharp geometric features and that our method improved on that criteria ; (*c* and *d*) initial and final target volume adjustment. Again we can note that the cells that need the biggest adjustment are those located near geometric features.

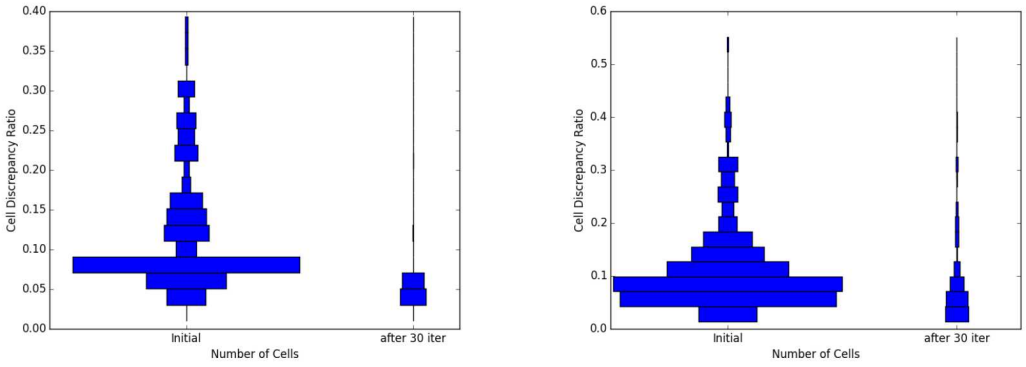


Fig. 9. Cell discrepancy ratio distribution in the initial and final M^O mesh (cells $c^I \in M^I$ that have $\frac{d(c^I)}{V(c^I)} < 0.05$ are not represented). Lower and narrower is better. (*a*) for the brick-cylinder with rotation angle 0, it confirms the difference seen between Figures 8.*a* and *b*. (*b*) same for the rotation angle 10 example.

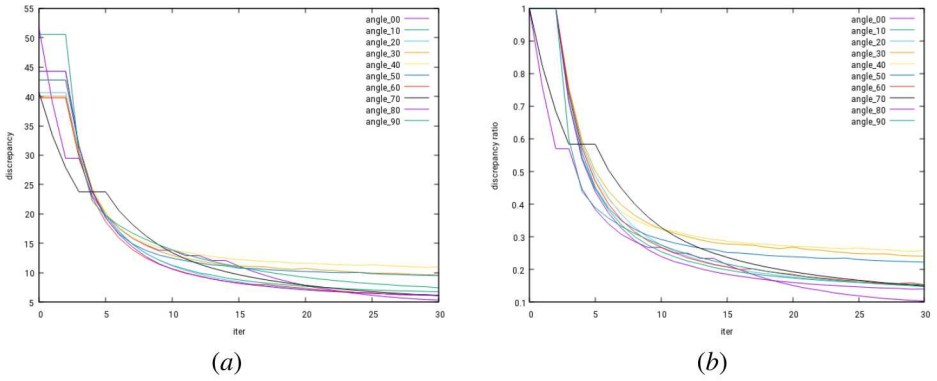


Fig. 10. (*a*) Discrepancy evolution for the brick-cylinder cases compared with iteration. Note that a step in the graph is indicative that node positions did not move due to elements reaching a minimum ϵ quality threshold. (*b*) Same data but with the ratio $\frac{d_{final}}{d_{init}}$.

6.2. Other examples

For additional validation of the proposed methods, we applied our approach on different types of input data. We give here the results from several examples based on three different input formats:

- CAD models (the v2_tweaked and the anc101 cases in Figures 11.*a* and 12.*a*) ;
- STL models (the lumbar and asteroid cases in Figures 11.*c* and *d*) ;
- Volume fraction models (the two-phase and the microstructure cases in Figures 11.*b* and 13.*a*).

Results from these examples are given in Table 3. Similar to the brick-cylinder case we note that discrepancies are reduced in all the cases.

6.3. Results analysis

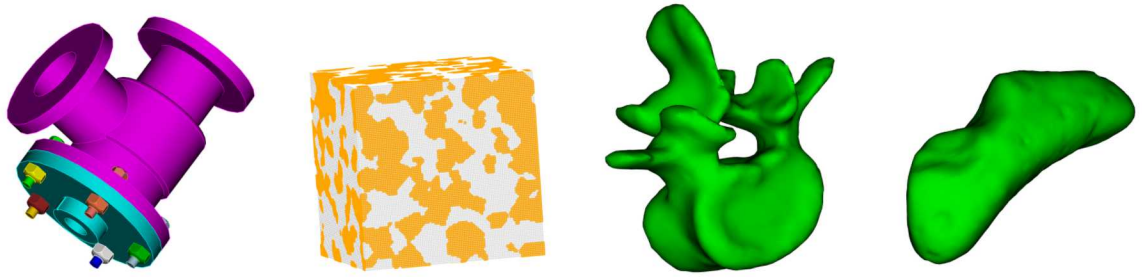
We note several observations from the preceding results. We focus on three specific features that required additional effort to apply or interpret the volume preservation procedures: namely, reloading node positions, fuzzy volume fractions and cell contribution error.

Table 1. Volume survey for the brick cylinder cases (the void material is excluded). We are interested here in evaluating the difference in material volume globally, as defined in Eq. 2. While volumes of the materials were already well preserved in the initial mesh our method still improves upon that.

angle (*degrees*)	$\frac{\Delta V_{init}(brick)}{V^I(brick)}$	$\frac{\Delta V_{final}(brick)}{V^I(brick)}$	$\frac{\Delta V_{init}(cyl)}{V^I(cyl)}$	$\frac{\Delta V_{final}(cyl)}{V^I(cyl)}$	$\frac{\Delta V_{init}}{V^I_{tot}}$	$\frac{\Delta V_{final}}{V^I_{tot}}$
0	8.79e-03	1.78e-05	-6.31e-03	-9.54e-06	8.25e-03	1.60e-05
10	2.26e-03	-1.39e-04	-2.07e-03	-5.27e-04	2.22e-03	2.24e-04
20	9.54e-04	-3.67e-04	-4.10e-04	-2.13e-04	8.34e-04	3.33e-04
30	7.99e-04	-5.49e-04	-9.21e-04	-6.77e-05	8.26e-04	4.43e-04
40	9.71e-04	-3.28e-04	-9.94e-04	3.81e-05	9.76e-04	2.65e-04
50	1.02e-03	-4.80e-04	-7.02e-04	9.02e-05	9.47e-04	3.94e-04
60	7.63e-04	-1.59e-04	-9.14e-04	-3.82e-05	7.96e-04	1.32e-04
70	9.62e-04	-2.39e-04	-1.61e-05	-1.97e-04	7.54e-04	2.30e-04
80	2.26e-03	-1.55e-06	-2.08e-03	-4.79e-04	2.22e-03	1.06e-04
90	-1.32e-03	-1.51e-04	4.19e-03	3.27e-04	1.95e-03	1.89e-04

Table 2. Discrepancy results for the brick cylinder cases. Columns 2 and 3 show the discrepancy as defined in Eq. 5. Columns 4 and 5 show the ratio of this discrepancy over the total volume of the materials; it expresses how "far" the M^O mesh is from the volume fractions carried by M^I. Column 6 is the ratio between the final and the initial total discrepancy where it is shown that our method has divided the initial discrepancy by a factor 4 to 10 depending on the case. Between columns 7 and 8 can be seen the improvement of the maximum discrepancy ratio per cell (the maximum of those values is 2, as seen in Eq. 7). In columns 9 and 10 are the minimum scaled jacobian before and after applying our procedure.

angle	d_{init}	d_{final}	$\frac{d_{init}}{V_{tot}}$	$\frac{d_{final}}{V_{tot}}$	$\frac{d_{final}}{d_{init}}$	$\max_{c\in M^I}(\frac{d_{init}(c)}{V(c)})$	$\max_{c\in M^I}(\frac{d_{final}(c)}{V(c)})$	$scaledJ_{init}$	$scaledJ_{final}$
0	51.6792	5.36181	4.03e-02	4.18e-03	1.04e-01	4.03e-01	1.28e-01	0.39	0.40
10	44.2667	6.76245	3.45e-02	5.28e-03	1.53e-01	5.66e-01	3.97e-01	0.26	0.20
20	40.6279	6.07868	3.17e-02	4.74e-03	1.50e-01	5.68e-01	3.42e-01	0.33	0.20
30	40.0265	9.62336	3.12e-02	7.51e-03	2.40e-01	5.02e-01	3.90e-01	0.31	0.20
40	42.72	11.0175	3.33e-02	8.60e-03	2.58e-01	5.42e-01	5.37e-01	0.26	0.20
50	42.7854	9.51502	3.34e-02	7.43e-03	2.22e-01	5.42e-01	4.79e-01	0.28	0.20
60	39.7635	6.09284	3.10e-02	4.75e-03	1.53e-01	5.02e-01	3.81e-01	0.31	0.20
70	40.7345	6.10349	3.18e-02	4.76e-03	1.50e-01	5.68e-01	2.08e-01	0.28	0.20
80	44.256	6.20802	3.45e-02	4.84e-03	1.40e-01	5.66e-01	3.89e-01	0.26	0.20
90	50.5343	7.44741	3.94e-02	5.81e-03	1.47e-01	4.08e-01	2.38e-01	0.28	0.20



(*a*) (*b*) (*c*) (*d*)

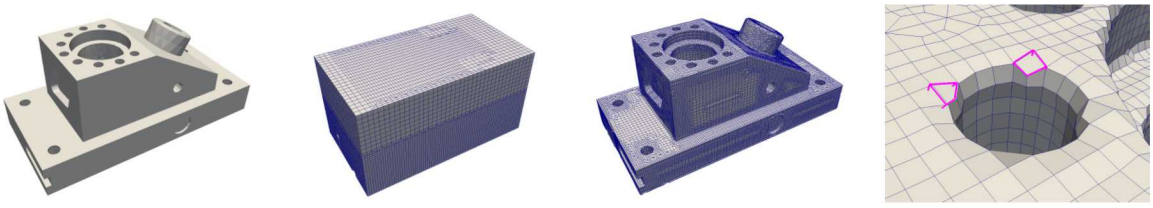
Fig. 11. Several input data. (*a*) CAD model for v2_tweaked ; (*b*) fraction presence for two_phase ; (*c*) STL for lumbar ; (*d*) STL for asteroid

Reloading node position. An initial implementation of our procedure moved and reloaded nodal locations when minimum mesh quality fell below a threshold ϵ scaled Jacobian. This implementation proved problematic, particularly in cases such as illustrated in Figure 12.*d*, where relocating nodes proved ineffective in improving local discrepancy. In these cases, relocating nodes can result in a few badly shaped elements. When this occurred, the procedure would reload previous positions of all nodes of the mesh. Subsequent iterations would compute new nodal offsets that would hopefully provide for improved mesh quality.

Table 3. Discrepancy results for several other examples.

angle	d_{init}	d_{final}	$\frac{d_{init}}{V_{tot}}$	$\frac{d_{final}}{V_{tot}}$	$\frac{d_{final}}{d_{init}}$	$\max_{c\in M^I}(\frac{d_{init}(c)}{V(c)})$	$\max_{c\in M^I}(\frac{d_{final}(c)}{V(c)})$	$scaledJ_{init}$	$scaledJ_{final}$
anc101	88975.4	7164.96	1.08e-02	8.73e-04	8.05e-02	5.88e-01	4.52e-01	0.20	0.15
v2_tweaked	212759	14622.9	2.72e-02	1.87e-03	6.87e-02	1.23e-01	5.75e-02	0.14	0.11
microstructure	142693	137804	1.61e-01	1.56e-01	9.66e-01	1.44e+00	1.40e+00	0.11	0.08
two-phase	94746100	55919300	9.47e-02	5.59e-02	5.90e-01	2.00	2.00	0.19	0.14
asteroid	12.914	1.87471	5.16e-03	7.48e-04	1.45e-01	6.25e-02	9.80e-02	0.23	0.17
lumbar	0.480773	0.057342	8.18e-03	9.76e-04	1.19e-01	4.36e-02	2.09e-02	0.20	0.15

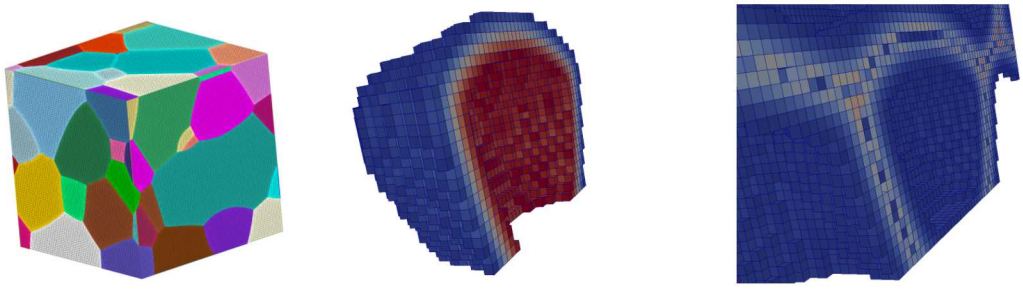
This proved to be a limiting factor of our method. As a consequence, we chose to identify such nodes and avoid moving them after several attempts. This limited the amount of discrepancy improvement at these nodes in favor of preserving a minimum cell quality, ϵ.



(*a*) (*b*) (*c*) (*d*)

Fig. 12. The anc101 example. (*a*) CAD model ; (*b*) M^I mesh, which is actually an adapted grid in this case ; (*c*) final M^O mesh ; (*d*) the two selected elements corner nodes should be moved outward in order to better follow the cylindric hole, but cannot be else the elements become non-convex.

Fuzzy volume fractions. In some cases, input volume fractions will not necessarily denote a sharp interface. Depending on the technology used while acquiring the data, the transition region between two materials may spread across several cells such as that illustrated in Figure 13.*b*. We observe this phenomenon in the microstructure test case illustrated in Figure 13.*a*. While the proposed method can still decrease the discrepancy it is more difficult to interpret the results since it will stay high whatever the changes brought to M^O. (see results for microstructure in Table 3) We note that this large transition between materials make for many additional non-zero discrepancy cells reducing the reported effectiveness of the procedure (see Fig. 13.*c*).



(*a*) (*b*) (*c*)

Fig. 13. The microstructure example. (*a*) M^O mesh ; (*b*) close-up of the volume fraction of one material. The cells represented are those with a volume fraction ranging from 0.1 to 0.9. In this case we observe that the transition is more than 4 cells wide (*c*) the discrepancy (cells $c^I \in M^I$ that have $\frac{d(c^I)}{V(c^I)} < 0.05$ are not represented), which is not limited to a width of one or two cells intersecting the interfaces between material, but spreads farther. The cells that actually intersect the interfaces between materials can in fact have a lower discrepancy than their neighbors.

Cell contribution error. We have observed that a cell of M^I, despite having a fraction of its volume composed of material m may have no part in moving the nodes of M^O at the interface of material m. Let c_i^I be a cell of M^I and m be a material where $f_{i,m} \neq 0$ and $c_i^I \cap M^0_{|m} = \emptyset$, in this case c_i^I has no contribution to the target volume of any cell of $M^O_{|m}$.

It is especially visible in the two-phase case : in Table 3 the maximum discrepancy ratio per cell $\max_{c^I \in M^I}(\frac{d(c^I)}{V(c^I)})$ is equal to the theoretical maximum of 2 (see Equation 7) because some small areas of one material do not appear in M^O.

7. Conclusion

We have proposed a post-processing procedure that improves upon mesh-first overlay grid methods by more accurately preserving volume of its reference geometry. The procedure was run on a variety of meshes built from STL models, complex CAD assemblies and volume fraction grids and proved to be effective in reducing the discrepancy criteria in all cases. In addition, geometric feature sharpness was improved compared to our input meshes.

While we have demonstrated the effectiveness of the procedures on overlay grid meshes, the proposed methods may conceivably be applied to any hexahedral mesh given a reference geometry. Where small deformations are applied to the reference geometric model, the hex mesh may be updated accordingly to preserve volume and density characteristics. We envision geometric deformations applied based on design changes or adaptively driven by an analysis code where discrete target densities are supplied to drive the volume preservation algorithm.

Although the proposed methods were demonstrated as a post-processing procedure applied to existing hexahedral meshes, future work will integrate the volume preservation algorithms directly into the Sculpt application. Sculpt uses a combined Laplacian-optimization-based smoothing technology, that while producing smooth material interfaces, may not directly preserve volume. Incorporating the volume preservation methods proposed in this work as part of Sculpt's smoothing procedures will result in more geometrically accurate hexahedral meshes.

References

[1] Y. J. Zhang, Geometric Modeling and Mesh Generation from Scanned Images, CRC Press, 2016.

[2] Openfoam user guide, snappyhexmesh, `https://cfd.direct/openfoam/user-guide/snappyhexmesh`, 2017.

[3] Distene, Volume meshing: Meshgems-hexa, `http://www.meshgems.com/volume-meshing-meshgems-hexa.html`, 2017.

[4] S. J. Owen, M. L. Staten, M. C. Sorensen, Parallel hex meshing from volume fractions, Engineering with Computers 30 (2014).

[5] S. N. Laboratories, Cubit 15.2 documentation, sweep, `http://cubit.sandia.gov/public/15.2/help_manual/WebHelp/cubithelp.htm`, 2017.

[6] Pointwise, Structured grid generation, `http://www.pointwise.com/pw/hex.shtml`, 2017.

[7] ANSYS, Introduction to ANSYS ICEM CFD HEXA, `http://www.ansys.com/Services/training-center/platform/introduction-to-ansys-icem-cfd-Hexa`, 2017.

[8] S. J. Owen, T. R. Shelton, Evaluation of grid-based hex meshes for solid mechanics, Engineering with Computers 31 (2015) 5–29.

[9] J. F. Shepherd, C. R. Johnson, Hexahedral mesh generation constraints, Engineering with Computers 25 (2009) 7–14.

[10] L. Maréchal, Advances in octree-based all-hexahedral mesh generation: Handling sharp features, in: B. Clark (Ed.), Proceedings of the 18th International Meshing Roundtable, Springer, Berlin, Heidelberg, 2009.

[11] M. Kucharik, R. V. Garimella, S. P. Schofield, M. J. Shashkov, A comparative study of interface reconstruction methods for multi-material ale simulations, Journal of Computational Physics 229 (2010) 2432–2452.

[12] A. M. Herring, O. Certik, C. R. Ferenbaugh, R. V. Garimella, B. A. Jean, C. M. Malone, C. M. Sewell, (u) introduction to portage, 2017. URL: `http://www.osti.gov/scitech/servlets/purl/1342852`.

[13] D. Powell, T. Abel, An exact general remeshing scheme applied to physically conservative voxelization, Journal of Computational Physics 297 (2015) 340–356.

[14] L. A. Freitag, On combining laplacian and optimization-based mesh smoothing techniques, in: TRENDS IN UNSTRUCTURED MESH GENERATION, 1997, pp. 37–43.

[15] D. Vartziotis, J. Wipper, Fast smoothing of mixed volume meshes based on the effective geometric element transformation method, Computer methods in applied mechanics and engineering 201 (2012) 65–81.