Computers & Graphics 114 (2023) 105–115

Contents lists available at ScienceDirect

Computers & Graphics

journal homepage: www.elsevier.com/locate/cag

Special Section on SMI 2023

Exploration of 3D motorcycle complexes from hexahedral meshes

Erkan Gunpinar [a], Marco Livesu [b,*], Marco Attene [b]

[a] Istanbul Technical University, Turkey
[b] CNR IMATI, Italy

ARTICLE INFO

Article history:
Received 10 May 2023
Accepted 5 June 2023
Available online 7 June 2023

Keywords:
Hexmeshing
Motorcycle graph
Domain decomposition

ABSTRACT

Shape decompositions that are guided by a motorcycle graph endow topological properties that are relevant for many engineering applications, such as T-spline fitting, shape compression and structured mesh generation. While for the surface case this is a widely studied and well-established construction, the concept of motorcycle graph was lifted to volumes only recently (Brückler et al., 2021). Due to this recent introduction, the generation of volumetric motorcycle graphs that fulfill application dependent criteria, such as minimal number of blocks or high approximation capabilities, is still an open problem. In this article we study and compare two alternative approaches to the computation of volume shape decompositions guided by a motorcycle graph. The proposed methodologies are designed to optimize alternative application-dependent quality criteria and, overall, perform better than prior art in most of the cases.

1. Introduction

Decomposing a given shape into an atlas of quadrilateral or cubic patches is an important step in many applications in graphics and engineering. In particular, non conforming partitions where patches are adjacent along a portion of an edge or a face allow to greatly reduce the overall number of primitives in the decomposition. For the surface case, patches are bounded by the arcs of the so called *motorcycle graph* [1,2], and the resulting decompositions are widely used as computational domain for higher order meshing [3], quadrilateral remeshing [4], mesh booleans [5], field-aligned surface mapping [6], texturing [7], and production of knitted models [8].

The concept of motorcycle graph was lifted from the surface to the volume setting only recently [9] and so far it was only exploited to perform robust quantization of volumetric Integer Grid Maps [10]. Considering the importance and variety of applications that are involved in the surface setting, motorcycle graphs promise to be equally useful in the volume setting, supporting a natural extension of the same applications for solid meshes.

In this work we first observe that the technique presented in [9] allows to compute *just one* among the (exponentially many) possible motorcycle graphs of a given mesh. Based on this observation, we propose novel algorithms that allow to systematically explore the space of solutions, permitting to find the decomposition that best suits the requirements of the application at hand. Due to this space size issue, an exhaustive search to find the absolute (global) optimum is often unfeasible. Rather, our approach identifies a practically good local optimum whose calculation requires a reasonable amount of time.

We explore the space of different 3D motorcycle complexes in a constructive manner. We propose two novel methods: 3D motorcycle complex enumeration and sheet swapping, which are based on serial fire growing metaphor introduced in [9]. Inspiration for these novel approaches comes from previous works in 2D [3,11], which explore the space of non conforming four sided partitions by segmenting an input quadrilateral surface mesh.

Our experiments confirm that the volumetric extension of these methodologies produce superior 3D motorcycle graphs than [9]. As an example, for the Stanford Dragon shown in Fig. 1 our best decomposition reduced the number of blocks from 220 to 187. More in general, when used for geometry compression purposes [12–14], replacing [9] with our approach leads to an average gain of approximately 3 bits-per-vertex (bpv), with peaks of 12 bpv for some test models. A detailed comparative analysis of our results can be found in Section 4.

2. Related works

This paper aims to compute non conforming coarse block decompositions that are aligned with the connectivity of a guiding hexahedral mesh. Hexahedral meshes are a prominent volume shape representation in graphics and engineering [15]. These decompositions are useful both to aid existing hexahedral meshing pipelines [10] and for downstream applications, where they

* Corresponding author.
E-mail addresses: gunpinar@itu.edu.tr (E. Gunpinar), marco.livesu@gmail.com (M. Livesu), marco.attene@ge.imati.cnr.it (M. Attene).

https://doi.org/10.1016/j.cag.2023.06.005

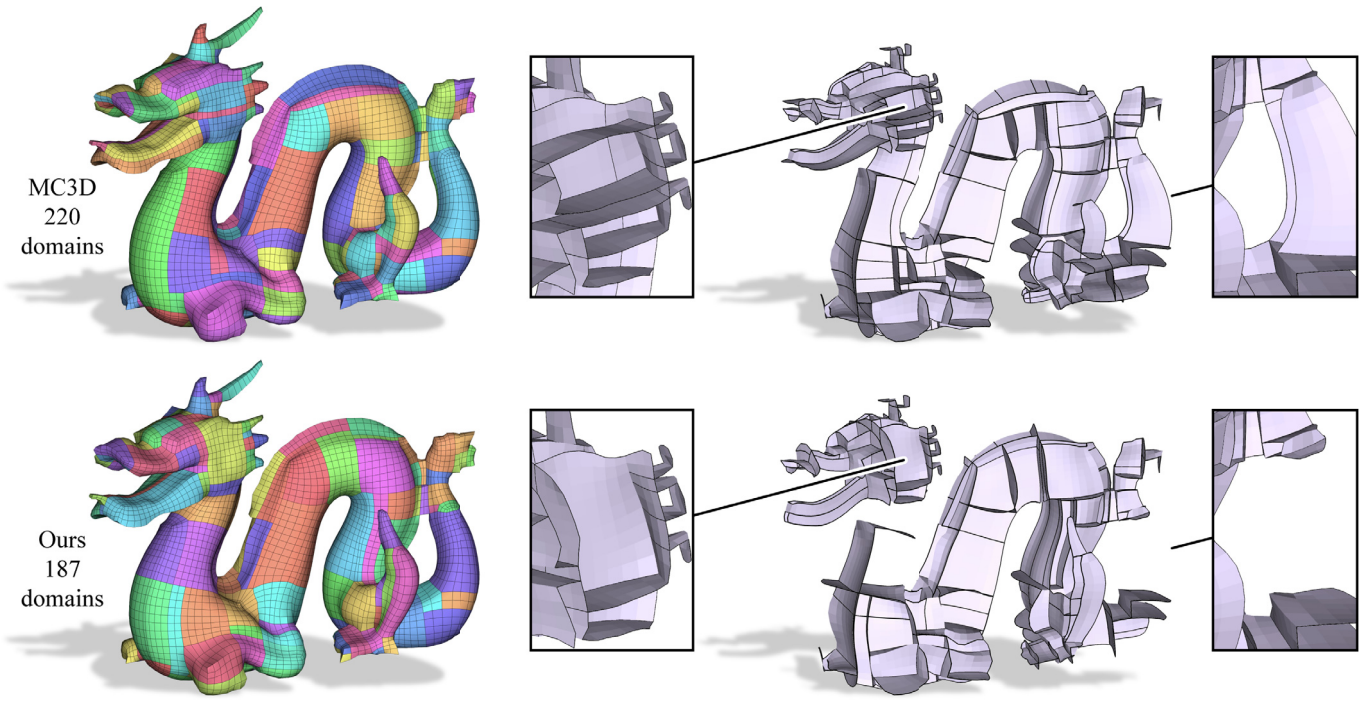

Fig. 1. 3D motorcycle complex as computed in [9] (top) versus an optimized complex computed by our method (bottom). The number of blocks is reduced by approximately 15%. Relevant differences are highlighted in the closeups on the right, where sheets separating the blocks are shown.

become a computational domain for the resolution of Partial Differential Equations (PDEs) with spline methods [16,17].

Literature proposes several approaches for the computation of hexahedral meshes, such as adaptive grids [18–21], polycubes [22–30], frame fields [31–37], sweeping along curves or membranes [38–40], and many others. Interactive approaches [23,41,42] or methods that occasionally introduce spurious non hexahedral cells [42–46] have also been explored.

Each of these methodologies is more or less capable to create meshes whose connectivity admits a *good* block decomposition for some application. The notion of good is application dependent, but typically relates to the number of domains in the decomposition, to the ability of each domain to accommodate a tensor product that approximates the target geometry well, to the block decomposition to compress a hexahedral mesh that is stored in a smaller amount of space, or to a combination of all these criteria. For example, grid-based methods are known to produce poorly structured meshes with high valence singularities [20]. As such, they are suitable for Finite Element Analysis (FEA), which operates on each element separately, but are unsuitable for spline methods such as Iso Geometric Analysis (IGA) [17], which demands the existence a coarse block decomposition. Conversely, frame field and polycube methods produce much coarser structures, obtained aligning singular vertices either at mesh generation time or in post processing [47–49].

Releasing the conformity requirement between adjacent blocks allows to greatly reduce the number of domains, also for complex meshes produced with grid methods. The recently proposed volumetric motorcycle graphs [9] can be used to generate such decompositions. Our work is positioned in this new line of research, improving the state of the art by permitting to explore a wider portion of the space of possible non conforming block decompositions, selecting the best suited for the application at hand based on a selected criteria such as minimization of number of blocks [9] and minimization of storage space [12–14]. Note that many science and engineering applications utilize high-resolution unstructured hexahedral meshes for modeling solid shapes for finite element simulations. These meshes need huge amount of space when stored in a raw format, and encoding their connectivity/geometry allows reduction of the storage space needed [12].

Exploration of combinatorial solution spaces. Regardless of the specific application, our algorithm is also loosely related to all methodologies that aim to make the exploration of a large combinatorial space efficient. This is indeed a fundamental computer science problem and an exhaustive discussion of this topic is beyond the scope of our paper. Perhaps not too distant from our applicative field are the topological methodologies used in [50], the branch-and-bound solvers used in [51,52] for layout embedding and tetrahedralization, and the hill climbing approach used in [26] to explore polycube segmentations. Our enumeration of the motorcycle graphs has also tight analogies with the beam search strategy used in [53] to decompose a complex shape into 3D printable chunks. Decomposition for digital manufacturing is indeed an application where similar problems often arise, and are solved greedily [54,55], via sampling of the solution space [56] or with other techniques. We point the reader to the recent tutorial at Eurographics for a comprehensive presentation of these and other methodologies [57].

3. Exploration of motorcycle graphs

Our method inputs a hexahedral mesh M and outputs a family of alternative partitions of it into non conforming cuboidal domains. We operate by systematically exploring the space of partitions so that, given an application-dependent quality metric, we can select the best suited for it as the best one among the partitions that were created by our algorithm.

In practice, a partitioning can be thought of as a labeling of the hexahedral elements in M, such that hexahedra having the same label belong to the same domain. Internal edges and faces of M that are incident to hexahedra with different labels define the boundaries of the cuboidal domains, jointly forming a Motorcycle Graph of M [9]. The motorcycle graph is a superset

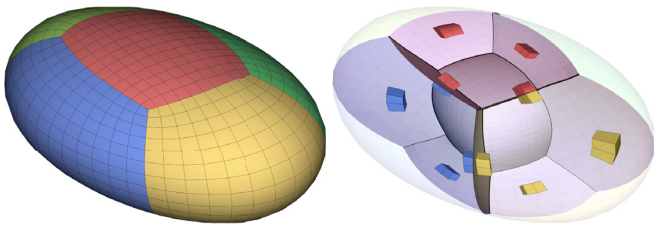

Fig. 2. Left: a hexahedral mesh with color coded elements that reflect its topological structure. Right: cutting sheets separating cuboidal domains are sets of contiguous quadrilateral faces whose incident hexahedra have different colors.

of the singular structure of M, which is fully contained in it. Consequently, hexahedral cells belonging to the same cuboidal domain are always arranged as a regular grid.

Considering that the space of alternative partitionings is exponential w.r.t. the number of singularities in the input mesh, exhaustively exploring all possible solutions to locate the global optimum is often not feasible. Our major difficulty is therefore to devise computationally feasible methods that allow to explore a portion of the space of solutions that is big enough and that likely contains either the global or a good local optimum. Note that our enumeration approach is remarkably different from the philosophy of [9], which greedily constructs a partitioning and then tries to locally modify its structure to reduce the number of blocks. By construction, such an approach can only explore the small fraction of the space of solutions located around the local optimum that was greedily identified, which can be arbitrarily bad for the application at hand.

In the remainder of the section we first explain how to generate a superset of the Motorcycle Graph starting from the singular structure of the hexahedral mesh M (Section 3.2) and then introduce our two novel strategies to navigate the space of solutions, one based on enumeration (Section 3.3) and the other based on swapping (Section 3.4).

3.1. Motorcycle graph superset

The topological structure endowed in the connectivity of a hexahedral mesh M is entirely defined by its singularities. A mesh edge is said to be *irregular*, or *singular*, if its number of incident cells is different from 4 in the interior or different from 2 at the boundary [15]. *Singular lines* in the mesh are chains of adjacent irregular edges that traverse the volume from one boundary vertex to another, form closed loops, or meet at an internal (irregular) vertex that is incident to three or more such chains. The number of hexahedra incident to each edge in a singular line determines its valence.

A singular line ℓ with valence v defines precisely v cutting sheets that emanate from it and, following the mesh connectivity, contribute to partition the volume into disjoint cuboidal components. Cutting sheets, which are propagated from all singular lines separately, can be easily computed with a flooding approach [9,49]: given a singular edge $e \in \ell$ and a quadrilateral face q incident to it, the whole cutting membrane can be computed by progressively conquering the quadrilateral faces that are adjacent to q through a regular edge and that are not faces of a hexahedron that is also incident to q (Fig. 2). For singular edges exposed on the surface, applying the tracing to boundary quads does not hurt but it can be avoided, because it simply reduces to flood a portion of the outer surface of M without contributing to its actual decomposition.

Cutting sheets emanating from different singular lines may intersect orthogonally at inner regular edges (in a topological sense). Tracing them all and accounting for their intersections yields a conforming cuboidal decomposition of M. This construction, called the *base complex*, can be thought of as the coarsest conforming hexahedral mesh derived from the connectivity of M [47,49] (i.e., each domain is a $1 \times 1 \times 1$ grid).

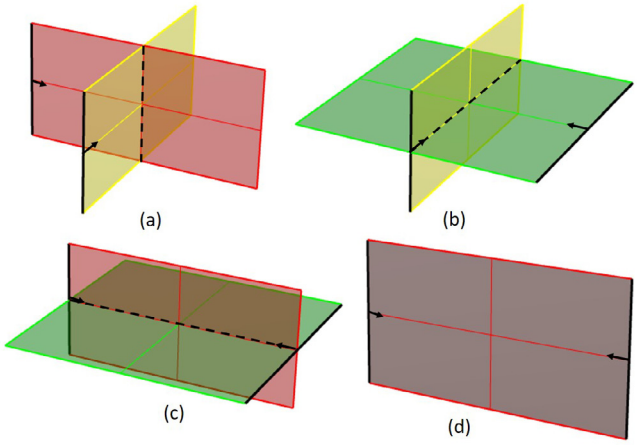

Fig. 3. Intersection types for two cutting sheets (in different colors). Black continuous and dashed lines depict singular lines and intersections, resp.

The base complex of M is a superset of all its motorcycle graphs, which are obtained by stopping the propagation of some of the cutting sheets at the intersection with other cutting sheets propagating from different singular lines [9].

As shown in Fig. 3 intersections may be of different types. Breaking ties at each intersection, deciding which cutting sheet should be propagated and which one should be stopped, defines the structure of the Motorcycle Graph and its suitability for downstream applications. This will be the subject of the two methodologies discussed in the next subsections.

3.2. Search space for motorcycle graphs

Our tools are designed to explore the space of motorcycle graphs that can be generated via a serial sheet insertion strategy [9], that is, cutting sheets are inserted one after the other while maximally expanding them until they hit a previously existing sheet. Compared to a simultaneous sheets insertion strategy, where two intersecting sheet may both be propagated past their intersection, the selected search space is obviously narrower. Fig. 4 illustrates possible cutting sheet configurations explored using the serial approach: *(a)* two orthogonal sheets intersect; *(b)* two coincident sheets (cyan and red) intersect a third one (yellow) orthogonally; *(c)* two orthogonal sheets (red and yellow) intersect a third sheet (green) orthogonal to both of them. In *(d)* we show configurations that cannot be obtained using our serial insertion strategy, because they involve partial propagation of a cutting sheet past an intersection line. Note that configurations in *(d)* are instead supported by [9] through a wall retraction mechanism, possibly obtaining solutions that are not in our search space. Nevertheless, propagating (partially or entirely) two sheets past their mutual intersection leads to denser decompositions that, as confirmed by our experiments, tend to be less optimal than configurations obtained with serial sheet insertion.

3.3. Enumeration

Now that the main decomposition tool (cutting sheet) has been introduced, it becomes easier to understand why the search

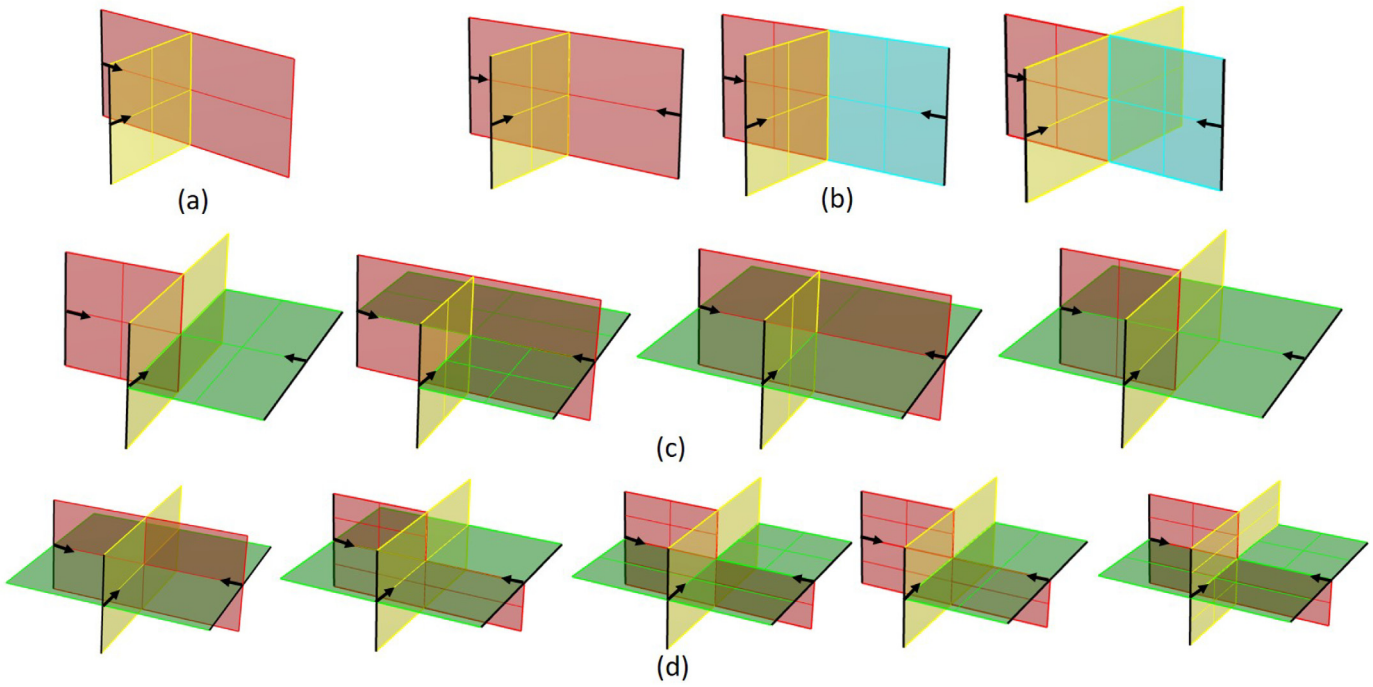

Fig. 4. Cutting sheet configurations: (a) two orthogonal sheets, (b) a sheet (yellow) orthogonal to two collapsing sheets (cyan and red), (c) two orthogonal sheets (red and yellow) and a sheet (green) not orthogonal to one of them (red). The configurations in (d) are not generated by the proposed methods in this paper.

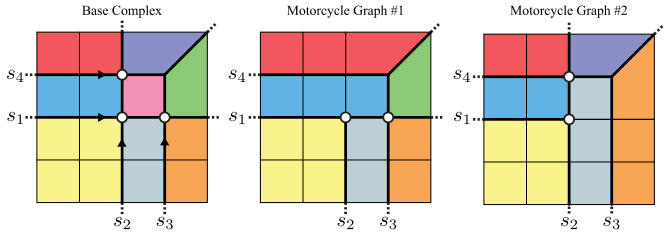

Fig. 5. Resolving intersections between separatrices at regular vertices (white circles) amounts to take a binary decision on which one should stop and which one should proceed. These decisions globally impact the resulting motorcycle graph. In the middle column s_1 stops s_2, as a result the domain changes and the intersection between s_2 and s_4 does not exist anymore. In the right column s_2 stops s_1, as a result the domain changes in a different way and the intersection between s_1 and s_3 does not exist anymore. These domain changes after each local binary choice design an exponential space of alternative solutions.

space for the motorcycle graphs of a hexahedral mesh M is exponential. To fix concepts we consider a simple 2D example shown in Fig. 5. Let us consider two cutting sheets s_1, s_2 that intersect along a chain of regular edges. Stopping one of the two at their intersection lines and letting the other continue to separate the mesh elements implies a binary choice (s_1 or s_2?). Now, let us consider other two cutting sheets s_3, s_4, one intersecting s_1 and the other intersecting s_2, both *after* the intersection between s_1 and s_2 occurs. If we go for s_1, then s_2 will not intersect with s_4 anymore. If we go for s_2, then s_1 will not intersect s_3 anymore. In both cases, the choice s_1 or s_2 *changes* the domain for the subsequent choices, thus leading to a combinatorial explosion in the number of alternative solutions. Not only this, but also the impact that each local decision has on the global final decomposition makes it very hard to take wise local choices to optimize a global desiderata, disqualifying greedy incremental approaches.

Conflict graph. To embrace all possible solutions in our search space, we start from a graph representation of the father of all motorcycle graphs, the base complex, and process it in order to enumerate all our candidate solutions. Hence we construct a graph G as follows:

- **Nodes:** we create a graph node for each cutting sheet emanating from a singular line, associating a unique identifier to it;
- **Arcs:** we connect two nodes n_i, n_j with an arc a_{ij} if the associated cutting sheets intersect orthogonally at a chain of regular mesh edges in one of the configurations shown in Fig. 3a, b, d. The configuration in Fig. 3c does not produce an arc in the graph because, as already observed in [9], the two associated cutting sheets do not block each other.

For aligned cutting sheets (i.e., sheets sharing same quadrilateral faces), two separate nodes (one for each) are created in G. The arcs of G encode all the possible conflicts between pairs of cutting sheets, which should be resolved with binary choices as discussed at the beginning of this section. It should be noted that only one arc is added for sheets intersecting each other multiple times, hence the local ordering between two intersecting sheets is always the same at each of their intersections. Starting from this graph we can now proceed enumerating all the possible solutions, which is done as described in the next paragraph.

Top-down strategy. Alternative motorcycle graphs are defined as leaves of a solution tree T, which we build incrementally with a top-down approach. We first initialize T with an isolated root node, associating the (un-partitioned) mesh M and the conflict graph G to it. The tree is then iteratively populated by expanding its leaf nodes as far as they can expand. New nodes in T also inherit a copy of the mesh M from their ancestors, enriching it with new cutting sheets that contribute to the volume decomposition. Eventually, each leaf node in T will contain an alternative decomposition of M (Fig. 6). As observed at the beginning of Section 3 decompositions are fully encoded by per element labels. Therefore, to minimize the memory footprint only per element labels are stored, while the original mesh and its topology are memorized only once.

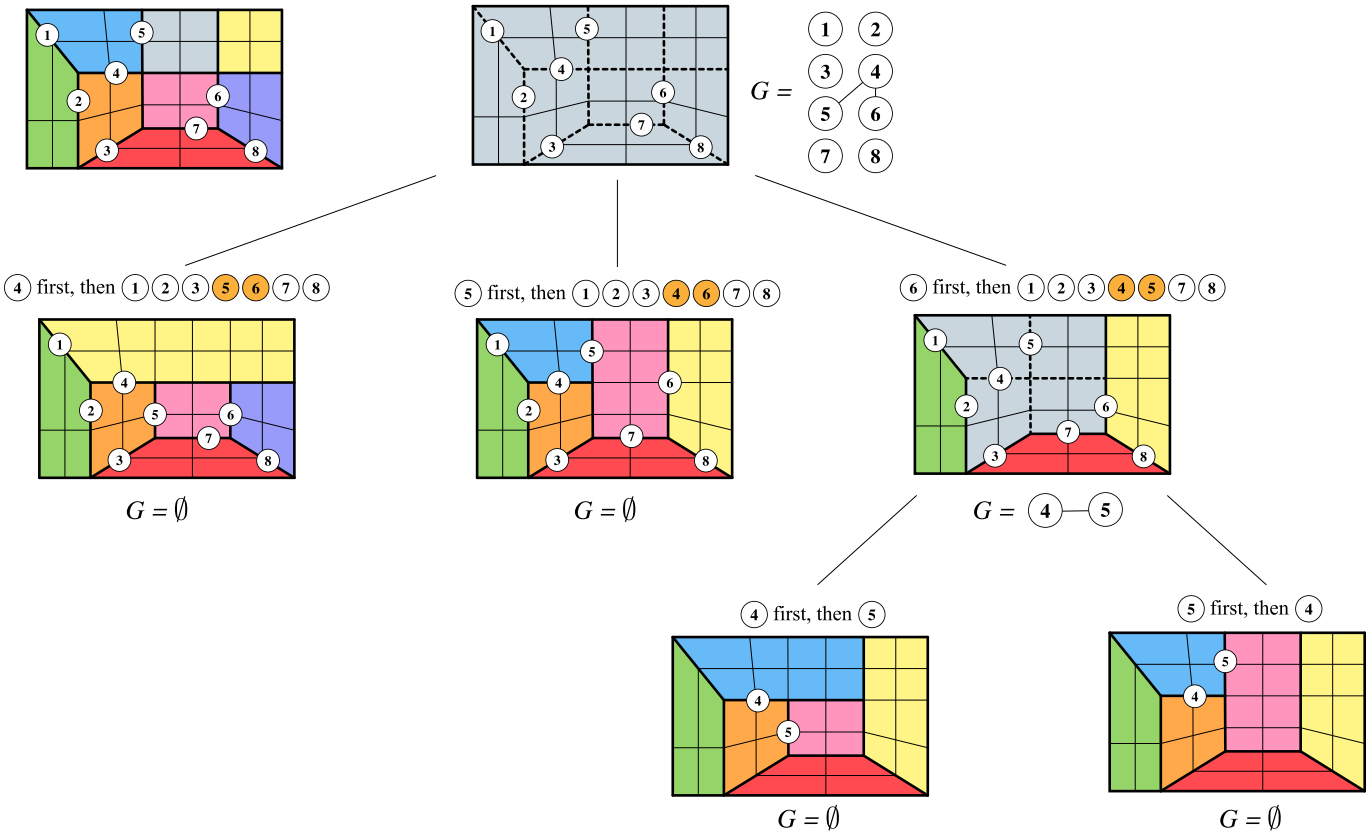

Fig. 6. Tree representation of the space of solutions for the base complex shown at the top left corner. Cutting sheets emanating from mesh singularities are numbered from 1 to 8. For each tree node, the conflict graph G encodes the sheets that intersect to one another. Sheets with at least one active conflict are explored in a dedicated branch, where they receive higher priority with respect to the other sheets in the graph. Nodes highlighted in orange are tested for mutual conflicts and are inserted only if no intersections are detected. Expanding tree branches until no conflicts are found yields the full space of solutions (leaves at the bottom of the tree). Note that duplicated solutions may exist (center and bottom right leaves).

Node expansion. Let us consider a generic leaf node n and the conflict graph G_n associated to it. Notice that since the global conflict graph G at the root of T encodes all conflicts in the base complex, the conflict graph G_n will be G at the root and a (possibly empty) subgraph of G at any other leaf node of T. All nodes in G_n that have at least one conflict (i.e. at least one incident arc in the graph) become children of n in T. For each child, we enrich the decomposition of M stored in n by tracing the associated cutting sheet first. Tracing is performed with the flooding approach described in Section 3.2, but flooding stops whenever a previously existing cutting sheet is intersected or the mesh boundary is reached. All the cutting sheets associated to the remaining nodes in G_n are traced afterwards. Sheets corresponding to isolated nodes in G_n are guaranteed intersection-free and are readily traced. Sheets corresponding to nodes in G_n with at least one conflict are traced only if they do not conflict with each other. In case conflicts are found, they are encoded in a novel conflict graph that will be associated to the current child. If no intersections are found, the conflict graph is set to $\emptyset$ and the current child will be a leaf. Since each node expansion in the tree resolves all conflicts with its associated sheet, the conflict graph of a child node is guaranteed to be a sub-graph of the conflict graph of its ancestor and will eventually become empty, ensuring convergence. The process stops when the conflict graphs associated to all leaf nodes of T do not contain any conflict. Intuitively, paths in the tree T define orderings between conflicting cutting sheets, meaning that sheets at the higher levels of T have priority w.r.t. the ones at the lower levels. Note that the so generated decompositions are not unique (Fig. 6). Duplicated solutions are removed in post processing. The final decomposition can be eventually chosen as the one that minimizes the application-dependent metric of choice. An example of exhaustive motorcycle graph exploration for a given hexahedral mesh is shown in Fig. 7.

Multiple conflict graphs. The full conflict graph G associated to the base complex is not necessarily connected, as the mesh may contain isolated clusters of intersecting sheets. If this is the case, one solution tree T is generated for each connected component separately. An exhaustive enumeration can still be computed by merging all leaf nodes with a cross-over blending, that is, taking each leaf from a cluster and merging it with each copy of all the leaves of all other existing clusters. Note that this costly operation is necessary only if one wants to explicitly create the whole space of solutions. Conversely, for the sake of detecting the best solution overall, one may exploit the fact that cutting sheets in different trees do not conflict to one another, therefore merging the best solution in each tree always yields the global optimum.

Pruning. The methodology described so far allows to fully enumerate all possible serial motorcycle graphs associated with a given input hexahedral mesh M. Considering the combinatorial explosion in the number of solutions, a full exploration is computationally feasible only for meshes with a simple singular structure such as the Bone model in Fig. 7. Similarly to recent beam search and branch-and-bound approaches [51,53] we often pruned the space of solutions, deciding to not expand all leaf nodes at all levels in order to reduce the amount of computation. There are different strategies that one can use to decide which nodes should be expanded and which ones not. In our

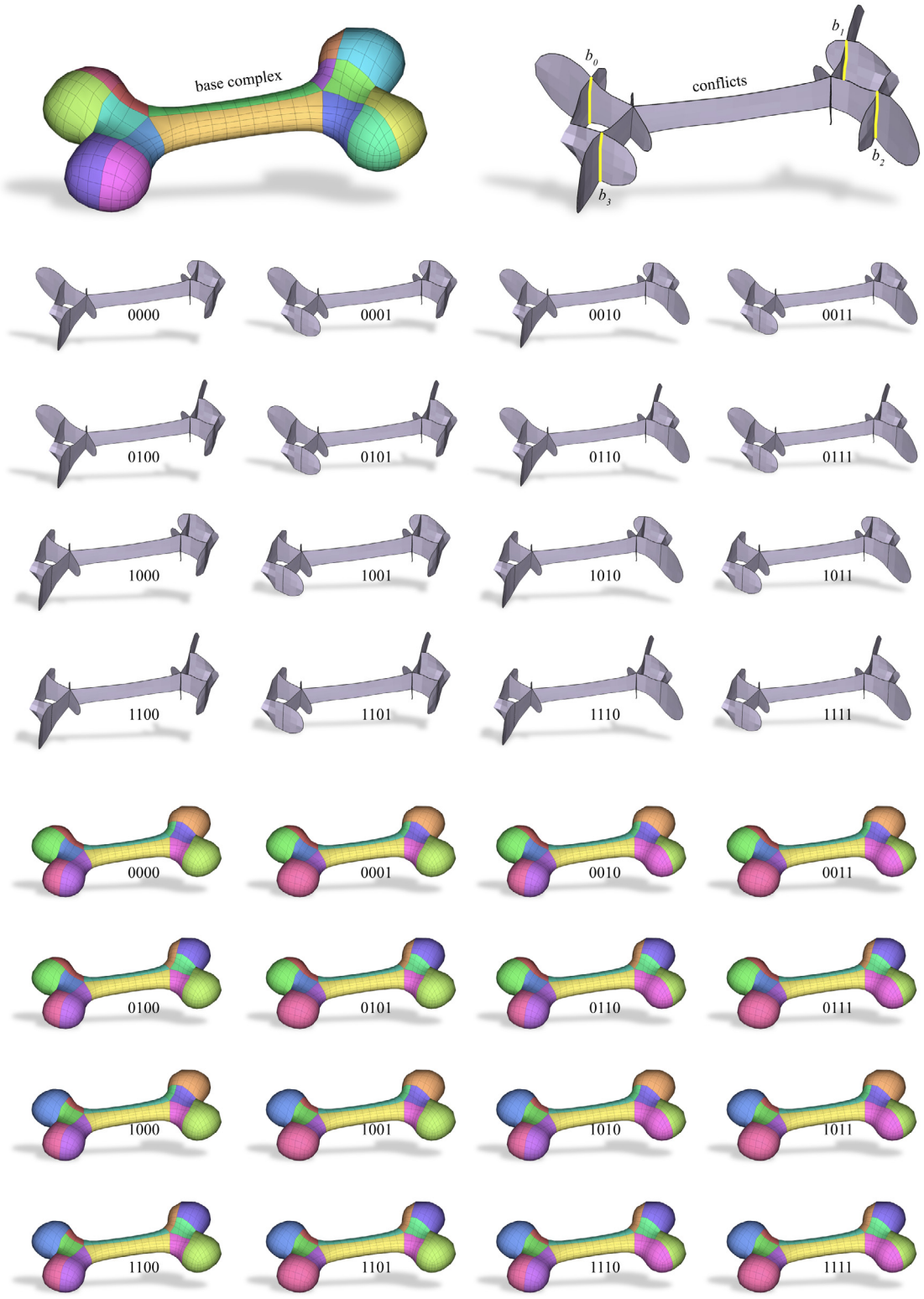

Fig. 7. Exhaustive enumeration of the motorcycle graphs of the Bone model. The base complex (top left) contains four conflicts (top right, yellow lines). Since conflicts are independent to one another, there exist 2^4 alternative solutions, here encoded with 4 bits (b_0, b_1, b_2, b_3) and shown in the figure both in terms of motorcycle graph (middle) and color coded volume decomposition (bottom).

Table 1
Model names.

Models	Names
1	2019 - Dual Sheet Meshing: An Interactive Approach to Robust Hexahedralization_bone
2	2019 - Selective Padding for Polycube-Based Hexahedral Meshing_Chamfer_L4
3	2019 - Selective Padding for Polycube-Based Hexahedral Meshing_Column
4	2012 - All-hex Meshing Using Singularity-restricted Field_fandisk
5	2020 - Cut-enhanced PolyCube-Maps for Feature-aware All-Hex Meshing_drill-trace1_1-hh-sat_size4
6	2012 - All-hex Meshing Using Singularity-restricted Field_hanger
7	2012 - All-hex Meshing Using Singularity-restricted Field_joint
8	2019 - Selective Padding for Polycube-Based Hexahedral Meshing_Bearing
9	2015 - Practical Hex-Mesh Optimization via Edge-Cone Rectification_block_out
10	2020 - LoopyCuts: Practical Feature-Preserving Block Decomposition for Strongly Hex-Dominant Meshing_cactus
11	2015 - Practical Hex-Mesh Optimization via Edge-Cone Rectification_cap_out
12	2019 - Selective Padding for Polycube-Based Hexahedral Meshing_Double_hinge_NH
13	2015 - Practical Hex-Mesh Optimization via Edge-Cone Rectification_dragon_out
14	2017 - A global approach to multi-axis swept mesh generation_Example_1
15	2012 - All-hex Meshing Using Singularity-restricted Field_impeller
16	2020 - LoopyCuts: Practical Feature-Preserving Block Decomposition for Strongly Hex-Dominant Meshing_impeller
17	2016 - All-hex meshing using closed-form induced polycube_joint-hex
18	2016 - All-hex meshing using closed-form induced polycube_kitten-hex
19	2020 - LoopyCuts: Practical Feature-Preserving Block Decomposition for Strongly Hex-Dominant Meshing_lever_arm
20	2012 - All-hex Meshing Using Singularity-restricted Field_rod
21	2016 - Polycube Simplification for Coarse Layouts of Surfaces and Volumes_hand-model_out
22	2019 - Selective Padding for Polycube-Based Hexahedral Meshing_Gear
23	2019 - Selective Padding for Polycube-Based Hexahedral Meshing_Wrench
24	2012 - All-hex Meshing Using Singularity-restricted Field_rockerarm
25	2017 - A global approach to multi-axis swept mesh generation_Example_3 (partial mesh)

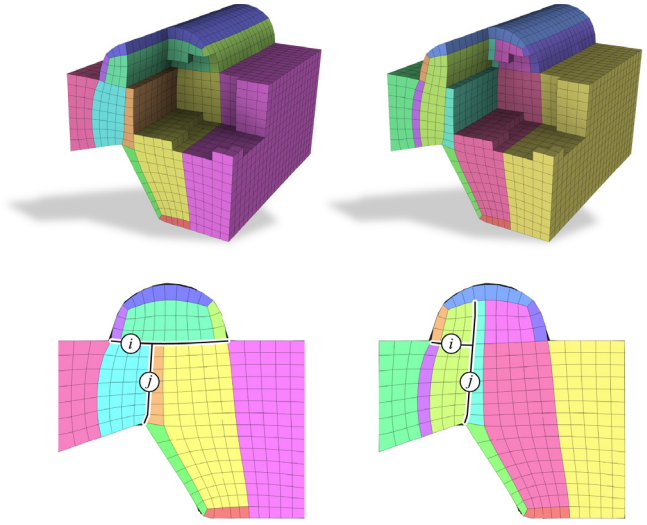

Fig. 8. Swapping the priority of sheets i and j yields two alternative decompositions. Our swapping strategy iteratively attempts local swaps, accepting only the moves that improve the quality of the decomposition, here measured in terms of number of cuboidal domains.

experiments we explored both a randomized approach and a preliminary evaluation of the optimization metric, discovering that randomization provides a better balance between computational cost and quality of the decomposition (Section 4).

3.4. Swapping

In addition to the enumeration strategy, we propose here an alternative greedy strategy, which is based on the intuition that modifying the binary decisions at the intersection between pairs of cutting sheets may increase the quality w.r.t. a target application. We call this strategy *swapping*, because indeed this operation amounts to swap the local priority between the two involved cutting sheets. In details, the swapping strategy is implemented as follows: we start from a randomly generated motorcycle graph. This can be selected among the ones produced by our enumeration algorithm or just created from scratch by growing all sheets together (akin [9]). We then list all the cutting sheet intersections it contains and go through this list. For each intersection, we attempt to locally switch the priority, extending the sheet that stopped and stopping the other one instead. If the so generated decomposition has a higher quality than the original one the move is accepted, it is reverted otherwise (Fig. 8). Quality assessment is performed according to the specific target application. In the experimental part we considered both the number of cuboidal elements and the compression rate achieved by the decomposition. Nevertheless, this strategy is also compatible with alternative quality metrics and applications. Note that in case the enumeration we compute is exhaustive, the global optimum is guaranteed to be found and the swapping strategy cannot improve the obtained result. In case pruning is used because the space of solutions was too large, swapping proved to effectively improve the quality of the output decomposition.

4. Results and discussion

Results of the methods proposed in this paper are obtained using 15 alternative hexahedral meshes retrieved from Hexalab [58] (Table 1). The following sub-sections describe cost functions used to drive the block partitioning, experimental details and processing times required by the algorithms.

4.1. Cost functions

While our method is agnostic to the specific metric used to evaluate the quality of a decomposition, we focused our attention on two practically relevant quality criteria: the number of blocks (E_n) and the extent of the compression rate that can be achieved on the decomposition (E_g). In both cases optimal decompositions are the minimizers of such metrics. E_n denotes the total number of blocks and is trivial to compute. E_g denotes the number of bits-per-vertex which are necessary to encode the decomposed mesh and is computed as follows.

Given an $M \times N \times T$ block with eight block-corners (blue dots in Fig. 9), the position of a vertex $p[i, j, k]$ (in red) is predicted

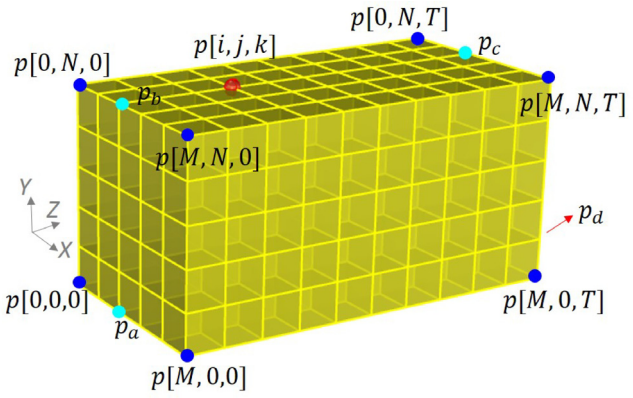

Fig. 9. 3D Coons-based prediction for a vertex $p[i, j, k]$ (red) in a $M \times N \times T$ grid (corners in blue). Auxiliary points used for the computation are cyan.

using 3D Coons patches [59]. Auxiliary vertices p_a, p_b, p_c, p_d (in cyan) are first found

$$p_a = p[0,0,0] + i\frac{(p[M,0,0] - p[0,0,0])}{M}$$
$$p_b = p[0,N,0] + i\frac{(p[M,N,0] - p[0,N,0])}{M}$$
$$p_c = p[0,N,T] + i\frac{(p[M,N,T] - p[0,N,T])}{M}$$
$$p_d = p[0,0,T] + i\frac{(p[M,0,T] - p[0,0,T])}{M}$$

then, the coordinates of $p[i, j, k]$ are computed as

$$p[i,j,k] = p_{ab} + k\frac{p_{cd} - p_{ab}}{T}, \tag{1}$$

where $p_{ab} = p_a + j\frac{p_b - p_a}{N}$ and $p_{cd} = p_c + j\frac{p_d - p_c}{N}$. To compute the geometry compression rate E_g the actual and predicted vertex coordinates are quantized using 12-bit integers by bounding the interval in which the coordinates lie [60]. For each vertex coordinate: (1) a bit z is computed, which indicates whether the two quantized values agree (i.e., perfect prediction or not). If they do not agree, (2) a bit s indicating the sign of the error (i.e., sign of the correction vector coordinate) and (3) the magnitude m of the error (i.e., absolute value for the correction vector coordinate) are calculated. The Shannon entropy E_z of z, E_s of s, and E_m of m are then computed as follows:

$$E_z|E_s|E_m = \sum_{i=0}^{Q-1} -(P_i \log_2(P_i)) \tag{2}$$

Here, Q denotes the number of values encountered, which is $2^1 - 1$ for E_z and E_s, while is $2^{12} - 1$ for E_m. P_i denotes the fraction of population (probability) of a value, i. $\log_2$ is the binary logarithm. Eventually, E_g is expressed in bit-per-vertex (bpv) and is defined as

$$E_g = \frac{3E_z + r\,(E_s + E_m)}{n_v} \tag{3}$$

where r is the number of vertex coordinates with wrong (imperfect) predictors among the three vertex coordinates and n_v is the number of mesh vertices.

4.2. Experiments

We tested hexmeshes in Table 1 with various combinations of the strategies described in Section 3. Specifically, we considered various versions of the enumeration algorithm, applying more or less aggressive pruning strategies to bound the number of solutions explored. We denote with *Ex* all such experiments, where x is an integer number denoting the maximum number of leaf nodes expanded at each tree level (chosen randomly among the existing leaves). Note that this corresponds to a beam search strategy with beam width x, with the only difference that the x expanded nodes are not the best ones according to some metric but are randomly chosen. For the swapping strategy, we applied it both to a randomly generated initial decomposition (Swap column in the tables) and we used it in combination with enumeration, selecting the best decomposition in the tree leaves and further processing it with swapping (E500 + Swap columns in the tables). For comparative analysis we also considered the 3D motorcycle complex algorithm (MC3D), which we used launching the original implementation[1] with a wall retraction option (i.e., "–mc –allow-selfadjacent –keep-singularity-walls"). Note that MC3D only allows to optimize for the number of blocks and it does not permit to minimize alternative quality metrics such as compression rate.

Table 2
Results of block partitioning based on number of blocks, E_n.

Model	MC3D	Swap	E100	E500	E1000	E5000	E10000	E500+Swap
1	12	12	12					12
2	11	11	11					11
3	17	18	17					17
4	22	20	23	22	20	21	21	20
5	51	60	73	69	65	64	59	54
6	22	21	22	24	24	21	21	21
7	28	28	41	28	28	28	28	28
8	66	58	68	62	63	60	60	58
9	98	98	100	99	99	99		99
10	41	43	45	44	43	44		43
11	85	84	105	114	110			95
12	35	33	37	36	36	34	34	33
13	220	191	246	230	241			194
14	153	154	198	183	189			143
15	124	102	102	101	104			101
16	148	146	146	146	156	146		146
17	22	22	24	24	22	22	21	21
18	57	60	100	62	60			60
19	53	54	67	64	62	62	62	54
20	27	22	34	30	37	22	22	22
21	71	71	75	75	74	73	72	71
22	45	39	41	41	41	39	39	39
23	30	29	29	29	29	29	29	29
24	119	123	130	133	135			113
25	168	167	175	175	175			169

Results based on E_n and E_g are arranged in Tables 2 and 3, respectively, which were considered in the experiments separately. Red values indicate the minimum (i.e., best) cost found for the decomposition. Perhaps not surprisingly, the combination of enumeration and swapping (E500+Swap) achieves the best result in the vast majority of cases, outperforming alternative approaches. When E_g is utilized as a quality metric, our method produces a better decomposition in 23 out of 25 cases, whereas MC3D was superior in just one case and equivalent in one other case. When E_n is the reference metric, our method performs better than MC3D in 16 out of 25 cases. Setting larger beam widths for the enumeration algorithm enables listing more motorcycle complexes, hence better solutions could be obtained. Note that since random pruning is used to limit the number of solutions computed, exploring a higher number of solutions is only likely to yield a better decomposition, but this is not guaranteed to happen. As a counterexample, for model 4 E1000 outperforms E5000 and E10000. Since MC3D does not allow to optimize for compression, its performances for E_g were worse than the ones

[1] https://github.com/HendrikBrueckler/MC3D.

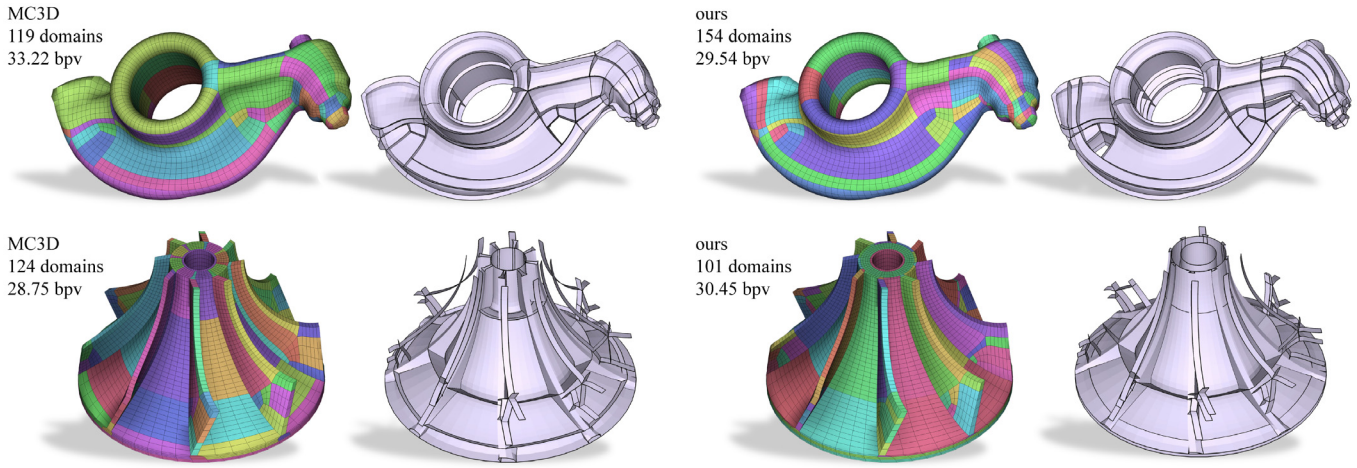

Fig. 10. Motorcycle graphs obtained with MC3D [9] and our swapping algorithm. MC3D does not currently support alternative metrics and it only allows to reduce the domains' count. For our method, for Rocker Arm (top) we optimized the geometric compression, expressed in bits-per-vertex. For the impeller (bottom) we optimized the number of domains.

Table 3
Results of block partitioning based on geometry compression rates, E_g, in bit-per-vertex (bpv).

Models	MC3D	Swap	E100	E500	E1000	E5000	E10000	E500+Swap
1	23.1	23.05	23.05					23.05
2	30.6	30.6	30.6					30.6
3	28.77	28.44	28.44					28.44
4	33.2	31.26	32.34	31.47	31.03	31.26	31.3	31.09
5	31.9	30.05	32.62	31.75	30.39	30.1	30.04	30.06
6	22.07	20.25	20.47	20.38	20.27	19.5	19.77	19.5
7	15.58	13.08	14.92	13.87	13.87	13.4	13.44	13.57
8	26.24	21.36	24.32	23.18	22.32	20.78	21.24	20.45
9	37.53	37.69	38.07	37.69	37.69	37.69		38.07
10	35.72	34.86	38.13	37.23	35.57	35.07		34.72
11	44.07	37.08	39.36	39.06	39.35			37.42
12	29.81	29.25	29.03	28.99	29.3	28.85	28.8	28.66
13	31.13	28.81	30.77	30.81	31.18			26.23
14	37.65	34.51	38.01	36.73	37.46			33.67
15	28.75	20.04	23.45	24.49	20.63			18.74
16	41.47	29.52	34.19	36.35	34.36	32.28		33.0
17	21.18	18.58	18.49	18.42	18.15	18.15	18.15	18.15
18	30.15	26.8	27.64	28.4	27.69			27.03
19	35.62	34.66	36.89	36.75	36.36	36.36	35.91	36.17
20	31.62	30.61	30.93	30.52	29.77	29.75	29.64	30.05
21	36.96	34.97	35.45	35.09	35.08	35	34.67	33.86
22	19.93	17.44	18.72	18.41	18.58	17.38	17.45	17.38
23	25.54	24.4	24.83	24.49	24.38	24.49	24.47	24.61
24	33.22	28.29	30.49	32.12	29.34			28.29
25	33.35	31.51	32.12	32.09	31.98			31.16

for E_n. Pictorial illustrations of our results are also reported in Figs. 1, 7, 10.

4.3. Computational time

Running times of the proposed algorithms are reported in Table 4. Swap_n and Swap_g refer to the fire swapping algorithm executed based on the cost functions E_n and E_g, respectively. Since Swap is purely greedy it has shorter processing time compared to the enumeration algorithm, which forms a tree graph structure and looks for the best solution in it. Running times highly depend on mesh resolution, which directly affects the number of faces participating in each cutting sheet. As a general rule, the higher the number of faces, the higher the computational cost hence the running time. An additional source of complexity is linked to the singular structure of the mesh. Meshes with a cleaner singular structure yield less cutting sheets and therefore design a smaller combinatorial space of solutions. Meshes with a dense singular structure (e.g., produced with grid-based approaches [20]) are much heavier to process.

4.4. Impact of randomization

Both the enumeration and the swapping algorithm employ randomization to prune the space of solutions or to perturb an existing decomposition. In this section we report on two experiments that aim to study the impact of randomization in the output solutions.

In the first experiment we launch the enumeration and swapping algorithms ten times on the Fan disk (model 4 in Table 1). This serves to evaluate how much variance in the results is introduced by randomization. For enumeration we considered a fixed beam width of 1000. For swapping we started from a randomly generated volume decomposition. For each method

Table 4
Running times (in seconds) for all our enumeration strategies.

Models	Swap_n	Swap_g	E100	E500	E1000	E5000	E10000
4	6	5	7	10	11	69	188
6	23	31	17	60	110	623	1917
8	155	224	116	364	710	9067	52218
12	7	9	19	96	192	1383	5036
17	4	5	5	19	28	152	424
20	17	26	53	295	553	3015	9911
21	288	347	279	839	1489	3108	60724
22	96	106	419	251	381	1939	4498
23	45	47	49	59	61	138	288

Table 5
E_n and E_g values for a fan disk model after multiple runs of the swapping and enumeration algorithms.

Run#	Swap_n	Swap_g	E1000_n	E1000_g
1	20	33.47	20	31.39
2	21	32.56	20	31.20
3	22	31.26	21	31.09
4	26	31.26	20	31.23
5	20	32.14	21	31.02
6	21	32.02	20	31.02
7	23	32.77	21	31.40
8	25	31.19	21	31.40
9	25	31.39	21	31.19
10	20	31.57	20	31.40

we attempted to minimize both the number of blocks (E_n) and the approximation error (E_g). As can be noticed from Table 5 results were very similar across the 10 runs, especially for the enumeration algorithm ($E_n \in [20, 21]$, $E_g \in [31.02, 31.4]$) which explores a good portion of the solution space and is therefore less affected by randomization. Higher but still acceptable (in our opinion) fluctuations were reported for the swapping method ($E_n \in [20, 26]$, $E_g \in [31.19, 33.47]$).

In the second experiment we evaluate the impact of randomization in the beam search strategy. We considered beam widths of 10, 100, 1000 and substituted our random leaf expansion policy with an alternative policy that is based on ranking leaves at each level according to the energy to be minimized. Results are summarized in Table 6 and can be compared with the ones in Tables 2 and 3 to assess differences with the random expansion strategy. Overall, it can observed that optimal decompositions obtained with a cost-based expansion strategy are only slightly better than the ones computed with a randomized approach. However, the cost to pay for this modest increment is a higher computational cost, which is due to the fact that the selected cost metric needs to be evaluated a high number of times, introducing a significant computational overhead.

4.5. IGA-based domain decompositions

As a final experiment with an alternative metric we considered the problem of computing a block decomposition for IsoGeometric Analysis (IGA), which prefers domains close to regular cubes. We fulfilled this requirement by using a novel quality metric, which is the ratio between the longest and shortest sides of each candidate domain, giving higher score to domains where this ratio is closer to one. This cost function was integrated into Swap and tested for models 4 and 8. Cost values for the generated block decompositions by Swap were, respectively, 86.71 and 247.82. On the other hand, these values were 92.33 and 531.6 when using MC3D. Fig. 11 shows the blocks of model 8 obtained using our Swap strategy and MC3D [9]. As can be noticed, MC3D introduced several elongated domains, whereas our method produced better-shaped (i.e., cubic) blocks.

Table 6
Results for cost-based enumeration (t : processing time in seconds).

Models	Cost	E10	E100	E1000
4	E_n	21	21	20
	t	5	11	37
6	E_n	21	21	21
	t	179	1247	7676
8	E_n	64	64	
	t	285	2104	
17	E_n	22	22	22
	t	19	65	389
20	E_n	22	22	22
	t	86	562	2594
21	E_n	82	78	
	t	384	2040	
22	E_n	40	39	
	t	458	4970	
23	E_n	30	29	29
	t	65	117	386

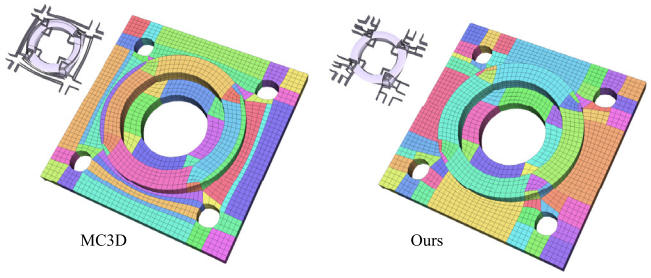

Fig. 11. Blocks obtained using MC3D and Swap for Iso-Geometric Analysis applications. Thanks to a customized quality metric our block decomposition discourages the presence of elongated domains, which are instead present in the MC3D result.

5. Conclusion and future works

This paper introduces two methods to explore 3D motorcycle complexes for a given hexahedral mesh. Inspired from similar works on exploration of motorcycle graphs for quadrilateral meshes [3,11] we introduced an algorithm for the exhaustive enumeration of all possible solutions, with pruning possibilities to reduce the computational effort and a greedy local swapping algorithm that allows to explore the solution space nearby a given starting solution that can be either randomly generated or come from a previous partial enumeration. The results of the proposed methods are validated and compared with those of Bruckler et al. [9], obtaining superior results in the majority of the cases and also opening for the use of customized application dependent quality metrics.

While we are satisfied with the results we obtained in our experiments, we also believe that the enumeration approach we implemented creates a useful basis to explore novel heuristics to prune the space of solutions in a smarter way, possibly obtaining even superior results. To this end, our future works will be focused on exploring this interesting research direction.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

Data will be made available on request.

Acknowledgments

This research was supported in part by Scientific Research Center of Istanbul Technical University, Turkey (grant No. FHD-2023-44808). Thanks are due to INdAM (Istituto Nazionale di Alta Matematica "Francesco Severi") and to colleagues at CNR-IMATI for helpful discussions. The authors would like to thank Hendrik Bruckler for helpful instructions on the installation of MC3D and Serhat Cam for installing and providing results of MC3D.

References

[1] Eppstein D, Goodrich MT, Kim E, Tamstorf R. Motorcycle graphs: Canonical quad mesh partitioning. Comput Graph Forum 2008;27(5):1477–86.
[2] Eppstein D, Erickson J. Raising roofs, crashing cycles, and playing pool: Applications of a data structure for finding pairwise interactions. In: Proceedings of the fourteenth annual symposium on computational geometry. New York, NY, USA: Association for Computing Machinery; 1998, p. 58–67.
[3] Gunpinar E, Moriguchi M, Suzuki H, Ohtake Y. Feature-aware partitions from the motorcycle graph. Comput Aided Des 2014;47:85–95.
[4] Campen M, Bommes D, Kobbelt L. Quantized global parametrization. Acm Trans Graph (Tog) 2015;34(6):1–12.
[5] Nuvoli S, Hernandez A, Esperança C, Scateni R, Cignoni P, Pietroni N. QuadMixer: Layout preserving blending of quadrilateral meshes. ACM Trans Graph 2019;38(6).
[6] Myles A, Pietroni N, Zorin D. Robust field-aligned global parametrization. ACM Trans Graph 2014;33(4):1–14.
[7] Schertler N, Panozzo D, Gumhold S, Tarini M. Generalized motorcycle graphs for imperfect quad-dominant meshes. ACM Trans Graph 2018;37(4).
[8] Wu K, Tarini M, Yuksel C, Mccann J, Gao X. Wearable 3D machine knitting: automatic generation of shaped knit sheets to cover real-world objects. IEEE Trans Vis Comput Graphics 2021.
[9] Brückler H, Gupta O, Mandad M, Campen M. The 3D motorcycle complex for structured volume decomposition. Comput Graph Forum 2021;41:221–35.
[10] Brückler H, Bommes D, Campen M. Volume parametrization quantization for hexahedral meshing. ACM Trans Graph 2022;41(4):1–19.
[11] Gunpinar E, Moriguchi M, Suzuki H, Ohtake Y. Motorcycle graph enumeration from quadrilateral meshes for reverse engineering. Comput Aided Des 2014;55:64–80.
[12] Isenburg M, Alliez P. Compressing hexahedral volume meshes. Graph Models 2003;65(4):239–57, Special Issue on Pacific Graphics 2002.
[13] Lindstrom P, Isenburg M. Lossless compression of hexahedral meshes. In: Data compression conference (Dcc 2008). 2008, p. 192–201.
[14] Courbet C, Isenburg M. Streaming compression of hexahedral meshes. Vis Comput 2010;26:1113–22.
[15] Pietroni N, Campen M, Sheffer A, Cherchi G, Bommes D, Gao X, Scateni R, Ledoux F, Remacle J-F, Livesu M. Hex-mesh generation and processing: A survey. ACM Trans Graph 2022.
[16] Martin T, Cohen E, Kirby M. Volumetric parameterization and trivariate B-spline fitting using harmonic functions. In: Proceedings of the 2008 ACM symposium on solid and physical modeling. Association for Computing Machinery; 2008, p. 269–80.
[17] Cottrell JA, Hughes TJ, Bazilevs Y. Isogeometric analysis: toward integration of CAD and FEA. John Wiley & Sons; 2009.
[18] Gao X, Shen H, Panozzo D. Feature preserving octree-based hexahedral meshing. In: Computer graphics forum, Vol. 38. Wiley Online Library; 2019, p. 135–49.
[19] Pitzalis L, Livesu M, Cherchi G, Gobbetti E, Scateni R. Generalized adaptive refinement for grid-based hexahedral meshing. Acm Trans Graph (SIGGRAPH Asia) 2021;40(6).
[20] Livesu M, Pitzalis L, Cherchi G. Optimal dual schemes for adaptive grid based hexmeshing. ACM Trans Graph 2022;41(2).
[21] Maréchal L. Advances in octree-based all-hexahedral mesh generation: handling sharp features. In: Proceedings of the 18th international meshing roundtable. Springer; 2009, p. 65–84.
[22] Gregson J, Sheffer A, Zhang E. All-hex mesh generation via volumetric polycube deformation. In: Computer graphics forum, Vol. 30. Wiley Online Library; 2011, p. 1407–16.
[23] Li L, Zhang P, Smirnov D, Abulnaga SM, Solomon J. Interactive all-hex meshing via cuboid decomposition. ACM Trans Graph 2021;40(6):1–17.
[24] Fang X, Xu W, Bao H, Jin. All-hex meshing using closed-form induced polycube. ACM Trans Graph 2016;35(4):1–9.
[25] Fu X-M, Bai C-Y, Liu Y. Efficient volumetric polycube-map construction. In: Computer graphics forum, Vol. 35. Wiley Online Library; 2016, p. 97–106.
[26] Livesu M, Vining N, Sheffer A, Gregson J, Scateni R. PolyCut: Monotone graph-cuts for PolyCube base-complex construction. Acm Trans Graph (Proc. SIGGRAPH ASIA 2013) 2013;32(6).
[27] Mandad M, Chen R, Bommes D, Campen M. Intrinsic mixed-integer polycubes for hexahedral meshing. Comput Aided Geom Design 2022;94:102078.
[28] Dumery C, Protais F, Mestrallet S, Bourcier C, Ledoux F. Evocube: A genetic labelling framework for polycube-maps. In: Computer graphics forum, Vol. 41. Wiley Online Library; 2022, p. 467–79.
[29] Guo H-X, Liu X, Yan D-M, Liu Y. Cut-enhanced PolyCube-maps for feature-aware all-hex meshing. ACM Trans Graph 2020;39(4).
[30] Huang J, Jiang T, Shi Z, Tong Y, Bao H, Desbrun M. ℓ_1-Based construction of polycube maps from complex shapes. ACM Trans Graph 2014;33(3):1–11.
[31] Li Y, Liu Y, Xu W, Wang W, Guo B. All-hex meshing using singularity-restricted field. ACM Trans Graph 2012;31(6).
[32] Nieser M, Reitebuch U, Polthier K. CubeCover– parameterization of 3D volumes. Comput Graph Forum 2011;30(5):1397–406.
[33] Palmer D, Bommes D, Solomon J. Algebraic representations for volumetric frame fields. ACM Trans Graph 2020;39(2).
[34] Liu H, Zhang P, Chien E, Solomon J, Bommes D. Singularity-constrained octahedral fields for hexahedral meshing. ACM Trans Graph 2018;37(4).
[35] Kowalski N, Ledoux F, Frey P. Smoothness driven frame field generation for hexahedral meshing. Comput Aided Des 2016;72:65–77, 23rd International Meshing Roundtable Special Issue: Advances in Mesh Generation.
[36] Solomon J, Vaxman A, Bommes D. Boundary element octahedral fields in volumes. ACM Trans Graph 2017;36(4).
[37] Corman E, Crane K. Symmetric moving frames. ACM Trans Graph 2019;38(4):1–16.
[38] Livesu M, Attene M, Patané G, Spagnuolo M. Explicit cylindrical maps for general tubular shapes. Comput Aided Des 2017;90:27–36.
[39] Livesu M, Muntoni A, Puppo E, Scateni R. Skeleton-driven adaptive hexahedral meshing of tubular shapes. Comput Graph Forum 2016;35(7):237–46.
[40] Gao X, Martin T, Deng S, Cohen E, Deng Z, Chen G. Structured volume decomposition via generalized sweeping. IEEE Trans Vis Comput Graphics 2015;22(7):1899–911.
[41] Takayama K. Dual sheet meshing: An interactive approach to robust hexahedralization. Comput Graph Forum 2019;38(2):37–48.
[42] Livesu M, Pietroni N, Puppo E, Sheffer A, Cignoni P. LoopyCuts: Practical feature-preserving block decomposition for strongly hex-dominant meshing. ACM Trans Graph 2020;39(4).
[43] Bukenberger DR, Tarini M, Lensch HP. At-most-hexa meshes. In: Computer graphics forum, Vol. 41. Wiley Online Library; 2022, p. 7–28.
[44] Gao X, Jakob W, Tarini M, Panozzo D. Robust hex-dominant mesh generation using field-guided polyhedral agglomeration. ACM Trans Graph 2017;36(4):1–13.
[45] Ray N, Sokolov D, Reberol M, Ledoux F, Lévy B. Hex-dominant meshing: mind the gap!. Comput Aided Des 2018;102:94–103.
[46] Sokolov D, Ray N, Untereiner L, Lévy B. Hexahedral-dominant meshing. ACM Trans Graph 2016;35(5):1–23.
[47] Cherchi G, Livesu M, Scateni R. Polycube simplification for coarse layouts of surfaces and volumes. Comput Graph Forum 2016;35(5):11–20.
[48] Protais F, Reberol M, Ray N, Corman E, Ledoux F, Sokolov D. Robust quantization for polycube maps. Comput Aided Des 2022;103321.
[49] Gao X, Deng Z, Chen G. Hexahedral mesh re-parameterization from aligned base-complex. ACM Trans Graph 2015;34(4).
[50] Verhetsel K, Pellerin J, Remacle J-F. Finding hexahedrizations for small quadrangulations of the sphere. ACM Trans Graph 2019;38(4):1–13.
[51] Layout embedding via combinatorial optimization. In: Computer graphics forum, Vol. 40. Wiley Online Library; 2021, p. 277–90.
[52] Marot C, Verhetsel K, Remacle J-F. Reviving the search for optimal tetrahedralizations. In: Proceedings of the 28th international meshing roundtable Zenodo, Buffalo, New York, USA. 2020.
[53] Luo L, Baran I, Rusinkiewicz S, Matusik W. Chopper: Partitioning models into 3D-printable parts. ACM Trans Graph 2012;31(6):1–9.
[54] Attene M. Shapes in a box: Disassembling 3D objects for efficient packing and fabrication. In: Computer graphics forum, Vol. 34. Wiley Online Library; 2015, p. 64–76.
[55] Fanni FA, Cherchi G, Muntoni A, Tola A, Scateni R. Fabrication oriented shape decomposition using polycube mapping. Comput Graph 2018;77:183–93.
[56] Muntoni A, Livesu M, Scateni R, Sheffer A, Panozzo D. Axis-aligned height-field block decomposition of 3D shapes. ACM Trans Graph 2018;37(5):1–15.
[57] Song P, Wang Z, Livesu M. Computational assemblies: Analysis, design, and fabrication. In: Hahmann S, Patow GA, editors. Eurographics 2022 - tutorials. The Eurographics Association; 2022, http://dx.doi.org/10.2312/egt.20221056.
[58] Bracci M, Tarini M, Pietroni N, Livesu M, Cignoni P. HexaLab.net: An online viewer for hexahedral meshes. Comput Aided Des 2019;110:24–36, https://www.hexalab.net/.
[59] Farin G, Hansford D. Discrete coons patches. Comput Aided Geom Design 1999;16(7):691–700.
[60] Pajarola R, Rossignac J. Compressed progressive meshes. IEEE Trans Vis Comput Graphics 2000;6(1):79–93.