Meso-Skeleton Guided Hexahedral Mesh Design

P. Viville[1] and P. Kraemer[1] and D. Bechmann[1]

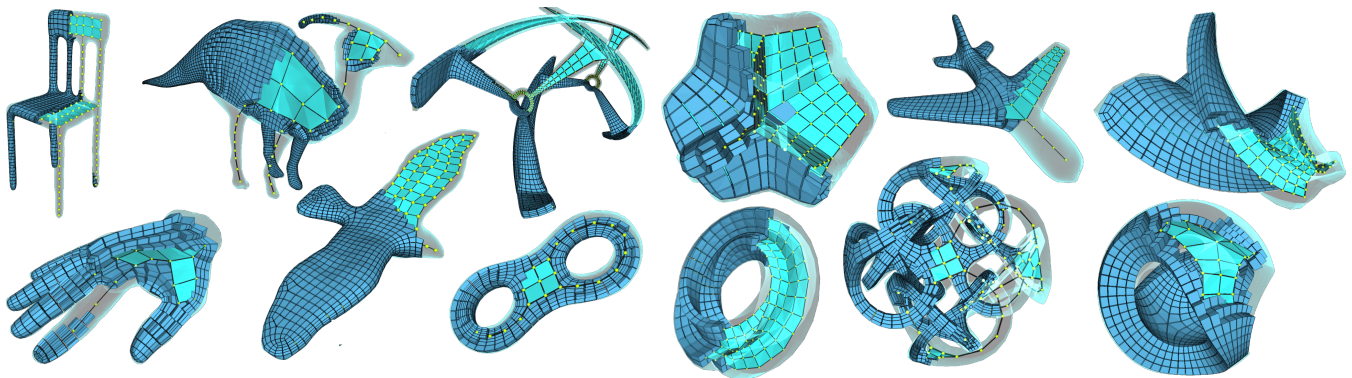[1]ICube, Université de Strasbourg, CNRS, France

Figure 1: *An overview of hexahedral meshes generated by our pipeline with the corresponding meso-skeletons used as support.*

Abstract
We present a novel approach for the generation of hexahedral meshes in a volume domain given its meso-skeleton. This compact representation of the topology and geometry, composed of both curve and surface parts, is used to produce a raw decomposition of the domain into hexahedral blocks. Analysis of the different local configurations of the skeleton leads to the construction of a set of connection surfaces that are used as a scaffold onto which the hexahedral blocks are assembled. These local configurations of the skeleton completely determine the singularities of the final mesh, and by following the skeleton, the geometry of the produced mesh naturally follows the geometry of the domain. Depending on the end user needs, the obtained mesh can be further adapted, refined or optimized, for example to better fit the boundary of the domain. Our algorithm does not involve the resolution of any global problem, most decisions are taken locally and it is thus highly suitable for parallel processing. This efficiency allows the user to stay in the loop for the correction or edition of the meso-skeleton for which a first sketch can be given by an existing automatic extraction algorithm.

CCS Concepts
• *Computing methodologies → Volumetric models; Mesh models;*

1. Introduction

The growing use of physical simulation in scientific research and industry has led to an increased demand for high-quality meshes as a simulation support. Hexahedral meshes, or hex meshes, are often favored for their numerical properties and the potential for optimization during simulation [OS16]. As a result, the number of research projects focused on the generation of hexahedral meshes has seen a significant increase in recent years [BRK*22]. However, the automatic generation of high-quality hexahedral meshes [Bla00] for arbitrary geometric domains remains an open problem [PCS*22].

General purpose hex meshing methods broadly fall within one of a few categories based on the used approach. These categories usually comprise of grid and octree based [Sch16, GSP19, LPC21], advancing fronts [KBLK14], user guided [LQS17, Tak19] or automatic [LPP*20, BGMC22, LZJ*17] block decomposition, volumetric parameterization [NRP11, LVS*13, LZS*21], and frame fields generation [KLF16, SRUL16]. Many of these methods are able to provide high quality hex meshes in a wide variety of cases. However, some of the classically cited drawbacks are the production of elements whose geometry is not always well aligned to the geometry of the boundary of the domain or the inability to guarantee purely hexahedral meshes as a result. This can be an issue
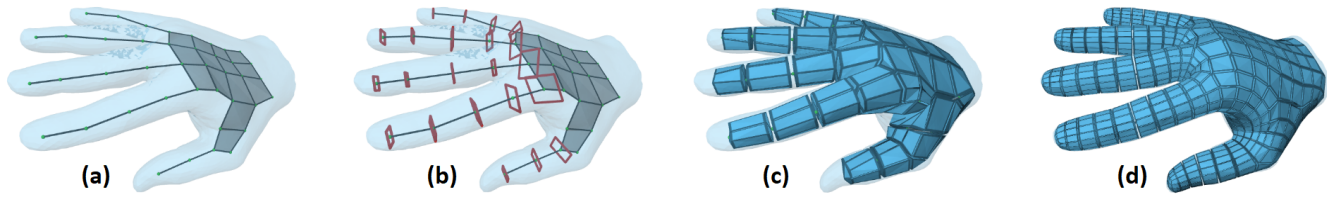
P. Viville & P. Kraemer & D. Bechmann / Meso-Skeleton Guided Hexahedral Mesh Design

Figure 2: *Application of the proposed algorithm to a hand model using its meso-skeleton. (a) The domain and its meso-skeleton composed of branches and leaflets. (b) Connection surfaces are built on joints of the skeleton. (c) The raw mesh is obtained by adding hex elements around branches and leaflets and sewing them using the incident connection surfaces. (d) An example of a final refined and optimized mesh.*

for some simulation tools that may need those guarantees. Furthermore, many of these methods involve significant computational costs and their computation times are often measured in minutes, even for simple shapes and even automatic methods sometimes fall back onto user input to correct and guide the process.

Restricting the targeted geometric domains to those sharing specific characteristics allows for the exploitation of some assumptions to propose specialized, hopefully more automatic and efficient mesh generation methods. Domains that can be faithfully captured by a curve skeleton are an example of such a specific characteristic. Research on generating hex meshes from skeletons is relatively limited. The method exposed in [LMPS16] is built upon an algorithm that was designed to generate quad surface meshes from a curve skeleton [ULP*15]. After fitting a cube on each branching point based on the branches directions, the output faces of these cubes are extruded along the branches. It produces generally good results but the propagation of local cuts on the faces of the branching cubes when multiple branches intersect the same face leads to the resolution of a global problem and leads to arbitrary choices in the presence of cycles.

Methods presented in [PMA21] and [VKB21] follow a more local approach to determine how the hexahedral elements should be connected in the vicinity of the branching points and do not require the resolution of a global problem. The generated mesh is fully hexahedral and its geometry is naturally well aligned with the surface of the domain. These methods are able to handle any type of branching point and their local nature enables efficient parallel processing. [DFOL22] also proposes a method to generate hex meshes from a curve skeleton following a similar approach. The size and the form factor of the elements prescribed by a simulation context are already accounted for in the cross section that is extruded to generate the hexahedra along the branches. These refinements and adaptations are rather seen as a post-processing task in [PMA21] and [VKB21].

Using a simplified curve skeleton as a proxy for the domain shape allows these methods to handle very complex tubular domains efficiently and to produce high-quality, surface-aligned hex meshes. However, they are limited to domains that are well captured by their curve skeleton. While very useful for example in the medical domain to generate volume meshes for structures like lungs or blood vessels [DFOL22], they handle a fairly narrow range of application domains. When the shape to mesh contains flattened areas, the quality of the results decreases rapidly, if a viable mesh can be obtained at all. In the example of Figure 3, generated us-

ing [VKB21], the elements built along the curve skeleton in the palm of the hand could be adjusted to fill the domain, but at the cost of severe deformations that will lead to very bad shaped or even inverted elements.

The medial axis, or medial object, of a domain bounded by a surface is well defined as the set of centers of inscribed empty spheres that touch the surface at more than one point. Methods presented in [PAR*20] and [Qua16] both use the medial object as a basis for decomposing the domain into blocks to build a hex mesh. Each block is then transformed into hexahedra using generic hex meshing methods, which leads to preserving their limitations. While carrying more information than a curve skeleton, medial objects are notoriously sensitive to noise hence making these methods more appropriate for smooth objects like CAD generated models.

In this paper, we propose a hex meshing algorithm that makes use of meso-skeletons (Figure 2), an approximation of the medial object composed of both curve and surface elements. Meso-skeletons can be a powerful support in hex-meshing as this skeletal representation already carries information about the topology and geometry of the domain. Our approach specifically extends the algorithm proposed by [VKB21] by adding the management of surfaces in the skeleton. As the meso-skeleton is better suited than a curve-skeleton to represent shapes with flattened areas, a wider variety of geometric domains can be addressed while preserving the simplicity and efficiency of the original algorithm.

Of course, the claimed simplicity and efficiency are closely related to the existence or the production of the used skeletons and their quality. The extraction, filtering and adaptation of curve skeletons is well studied and several known methods produce quality results [TDS*16]. There are not many existing studies aiming at the creation of a meso-skeleton of a given shape. Such a struc-
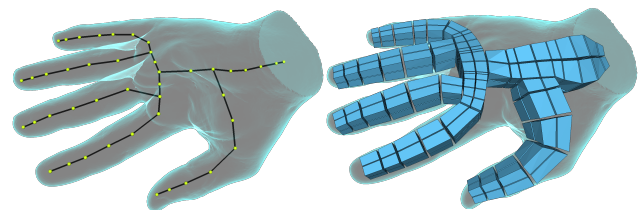

Figure 3: *Methods that use a curve skeleton to produce hexahedral meshes are not well suited to domains that have non-tubular parts.*

ture appears as an intermediate step of the curve skeleton retrieval method presented in [TAOZ12], but the extraction of a proper complex composed of curve and surface parts is not clearly described. [YSC*16] and [LCLJ10] offer methods to generate either curve or meso-skeletons with reliable, yet noisy results. The recent methods proposed in [DLX*22] and [LWS*16] are clearly targeted towards this goal and can already produce usable results in some configurations.

As we will see in the following, our method relies on a given clean meso-skeleton in which the surface parts are composed of quad faces. The automatic extraction of such a meso-skeleton is not yet achieved reliably enough by existing methods. However, given recent developments in that field, we believe that reliable methods to produce the expected meso-skeletons will be available in the near future. If a suitable skeleton cannot be obtained automatically, the efficiency of the proposed algorithm allows the user to update the meso-skeleton and obtain the corresponding hex-mesh in an interactive loop. As we think some oversight by the user in the production of the meso-skeleton is not something to be avoided at all costs (no automatic tool can guess and meet all end user constraints and expectations), the interactive edition tools that we built will still be useful. Most of the results presented here use hand-made or at least hand-adapted meso-skeletons.

The remaining of the article is organized as follows. In the next section we briefly describe the main steps of the algorithm. Section 3 is dedicated to the details of each of these steps, focusing on the management of the surface parts of the meso-skeleton. Some results are presented in Section 4. In Section 5 we discuss the limitations of the current approach. The conclusion summarizes the proposed algorithm and gives some insights for future work.

2. Main steps

Our proposed algorithm is built upon the pipeline described in [VKB21] that took a surface and its curve skeleton as input to generate the hex mesh. We extend here the capabilities of this approach by allowing the skeleton to also have surface parts.

Let us first introduce the terminology used to denote the different parts of a meso-skeleton (see Figure 4 for an example). As in [VKB21], within the curve parts of a skeleton, we call *extremity* a vertex of degree 1, *joint* a vertex of degree 2, and *branching point* a vertex of any higher degree. Extremities and branching points are the ends of *branches* along which only joints can be found. New terms are needed to describe the configurations that involve surface parts. We call *leaflet* (depicted with different colors in Figure 4(b)) a 2-manifold quad mesh bounded by edges of degree 1 or higher than 2. Edges of degree higher than 2 are called *fan edges*. Vertices incident to 2 leaflets are called *LL-joints*. Vertices incident to 1 branch and 1 leaflet are called *BL-joints*. Vertices incident to more than 2 leaflets or branches are considered as regular *branching points*.

The original pipeline consisted of 7 main steps that produce a coarse hex mesh. Most of these steps have to be adapted in order to manage the surface parts of the skeleton:

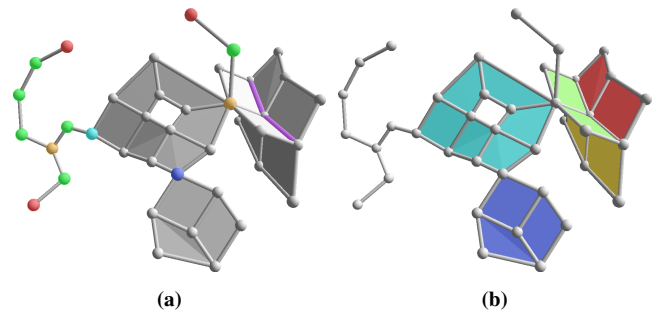1. **Skeleton adaptation**. The sampling density of the skeleton is

(a)   **(b)**

Figure 4: *Classification of cells in a meso-skeleton. (a) Extremities are in red, joints in green, branching points in yellow, LL-joints (Leaf-Leaf joints) in blue, BL-joints (Branch-Leaf joints) in cyan, and fan edges in purple. (b) Each of the five leaflets is depicted with a different color.*

adapted locally to the geometry of the shape. The length of the edges and form factor of the faces of the meso-skeleton will have a direct influence on the generated hexahedra in the raw mesh.
2. **Skeleton analysis**. The branches, leaflets and all types of branching points and joints are counted and characterized. This allows for a potential anticipated resources allocation, as the structure of the generated hex mesh is fully known after this step.
3. **Connection surfaces construction**. A quadrilateral partition of a sphere is constructed on each branching point and joint of the skeleton based on the directions of the incident branches as well as leaflets. These partitions are called *connection surfaces* and serve as a scaffold onto which the hexahedral mesh is built.
4. **Geometry propagation**. Geometric constraints present on the connection surfaces at branching points are propagated along the branches to the connection surfaces at the joints and extremities.
5. **Blocks insertion**. *Blocks* composed of hexahedra are inserted on each edge of the branches. New types of blocks are created on each face and boundary edge of the leaflets. The blocks are registered within their incident connection surfaces and the skeleton. After this step, each edge of the connection surfaces and each edge of the leaflets is associated with two hexahedra.
6. **Volumes sewing**. The pairs of hexahedra associated with each edge of the connection surfaces and leaflet edges are sewn together.
7. **Geometry transfer**. The geometry encoded in the skeleton and the connection surfaces is transferred to the hexahedra, completing the raw hex mesh.

This pipeline produces a rather coarse hex mesh. It can then be padded, refined and optimized as required by the user, for example to better fit the surface of the domain. This results in a mostly regular hex mesh (a regular edge has 4 incident hexahedra). The irregularities are localized around certain skeleton branching points, fan edges and irregular leaflet vertices. The mesh is aligned with the surface of the domain, as induced by the underlying meso-skeleton. Most steps only involve independent decisions localized to a specific part or type of cell in the mesh, allowing straightforward par-

allel processing. Figure 2 illustrates an example of the application of this algorithm on a model of a hand.

3. Algorithm details

3.1. Skeleton adaptation

The first step consists in adapting the sampling and connectivity of the skeleton. The density and shape of the cells of the skeleton strongly influence the result of the whole algorithm, as the edges and faces of the meso-skeleton will form the basis of the hexahedra that will be generated in the following steps. The goal of this step is to obtain the best starting point, depending on the application context. For example, one can aim at the generation of as regular as possible cube-shaped hexahedra. The adaptation of the skeleton will then use the local radius of the shape to determine the sampling density.
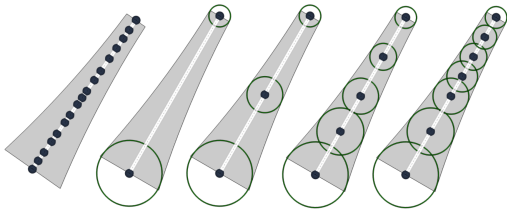

Figure 5: *An oversampled skeleton branch is simplified and recursively subdivided. The local radius and the proximity of the neighbours are used as a termination condition of the process. As a result, the length of the edges in the resampled branch is adapted to the local radius.*

Along branches, if an automatic skeleton extraction technique has produced an oversampled skeleton, a radius aware procedure inspired by [LMPS16] and used by [VKB21] (Figure 5) can be applied. First, an arc-length parameterization of the branch is computed and the branch is simplified to a single segment. This segment is then recursively cut in two by inserting a vertex in the middle. Inserted vertices are placed at the position corresponding to their parametric coordinate in the original curve. The radius of the shape at the vertices position is estimated and this process continues until the radius at an inserted vertex overlaps with that of its two neighbors. At the end of the process, the lengths of the edges of the skeleton depend locally on the radius of the shape. Each branch of the meso-skeleton can be processed independently.

The following steps of our algorithm need leaflets to be composed of quadrilateral faces and their vertices should be as regular as possible, i.e. have 4 incident edges. Indeed, as illustrated in Figure 6, irregular vertices in the leaflets will eventually lead to irregular edges in the generated hex mesh. As stated in the introduction, only few methods are currently able to automatically extract meso-skeletons [TAOZ12, DLX*22, LWS*16], and their surface parts are generally composed of triangles. Quadrilateralization methods, such as the ones proposed in [OSCS99, RLS*12] can be used to transform these triangle meshes into quad meshes. Irregularities can be removed using local regularization operators like proposed in [DSH20]. If the density of the triangle mesh is already adapted to the local radius of the shape, which can be achieved

by classical surface remeshing algorithms, those methods are able to maintain this adaptation in the produced quad mesh. However, variable density in the leaflet sampling will inevitably lead to some irregular vertices.
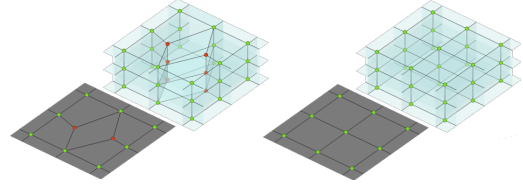

Figure 6: *Irregular vertices within a leaflet (in red) will lead to irregular edges in the resulting hexahedral mesh, reducing the maximum achievable quality of the mesh.*

Another way to obtain leaflets that meet the requirements is to ask the user for input. We designed an interactive tool that lets the user model a meso-skeleton for a given domain, either starting from scratch or from a given skeleton with or without surface parts. The main operations are extrusion and local connectivity modifications. As the last steps of our method include geometry optimization and fitting of the hex mesh to the domain surface, the geometry of the skeleton does not actually need to be particularly accurate with respect to the underlying medial object.

3.2. Skeleton analysis

The second step of the algorithm involves analyzing the meso-skeleton to determine its composition and how the branches and leaflets are connected. This analysis is done through a flooding process.

Branches are extracted by flooding edges, starting from any edge of degree 0 (not incident to any face), through *joints*, until any other kind of vertex are met (*branching point*, *LL-joint* or *BL-joint*). Leaflets are extracted by flooding faces, starting from any face, through edges of degree 2, until boundary edges or *fan edges* are met. To ease the subsequent steps of the algorithm, the different kinds of vertices described in Figure 4 are also identified.

The exact number of construction blocks needed to build the raw mesh can be determined at this step, allowing for pre-allocation of resources. The irregularities of the raw mesh produced at the end of the pipeline are also known at this step. Details about the regularity of the obtained mesh are given in the results section.

3.3. Connection surfaces construction

In this step, connection surfaces are constructed on all *branching points* and *joints* of the skeleton. They serve two purposes: encode the way hexahedra created on the incident branches and leaflets will be connected together and store geometrical information. In essence, they are the scaffold upon which the hexahedral mesh is built.

The main difficulty in skeleton-driven hex mesh generation is managing the connectivity around branching points. If the branches are composed of a single extruded hex, we are left with a set of
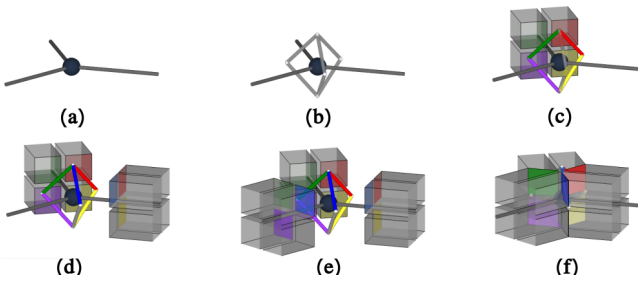
Figure 7: *Connection of hexahedral chunks around a branching point using a quadrilateral connection surface. (a) A vertex incident to 3 branches. (b) The connection surface composed of 3 quadrilaterals. (c, d, e) Hexahedra of incident chunks are associated with the edges of their respective quadrilateral. (f) Pairs of hexahedra are connected together. (Figure 4. of [VKB21])*

incoming quads around branching points. Filling the polyhedron formed by these quads with hexahedra is a very complicated problem in the general case [VPR19]. To address this issue, an alternative approach is proposed in [VKB21]. Along branches, a chunk of four hexahedra is created on each edge (Figure 7 c). Using blocks of four hexahedra minimizes the introduction of irregular edges along branches in the final hex mesh. On each branching point and joint, the connection surface, a polyhedron composed of one quad per incident branch is constructed. The four incoming hexahedra of each incident branch are associated with the four edges of their corresponding quad face in the connection surface. Finally, the pairs of hexahedra associated to the edges of the connection surface are sewn together. This process, illustrated in Figure 7 for a branching point of degree 3 can be generalized to branching points of any degree.

The connectivity and the geometry of the connection surface is obtained by using a deterministic sphere partitioning process defined by [VKB21], illustrated in Figure 8, that outputs one quad per incident branch. This approach is still suitable in the context of meso-skeletons, for branching points that are not incident to any leaflet (only incident to branches).

Branching points that are incident to leaflets are processed using a similar approach. Each incident leaflet is considered as a unique virtual branch with its average direction, as illustrated in Figure 9.
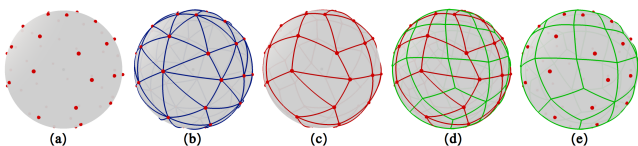

Figure 8: *Partitioning a sphere in a given number of quads (one per incident branch). The method proposed in [VKB21] starts with a set of points on the sphere (a), creates the Delaunay Triangulation of the points (b), updates the connectivity using edge flips, edge additions and deletions until each vertex has 4 incident edges (c), then takes the dual of this mesh (d) which is the final quad mesh (e).*
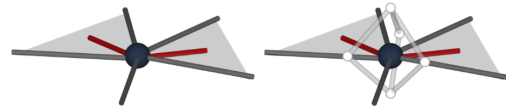

Figure 9: *On branching points that are incident to both branches and leaflets (or leaflets only), the connection surface is constructed by considering the average direction of each leaflet.*
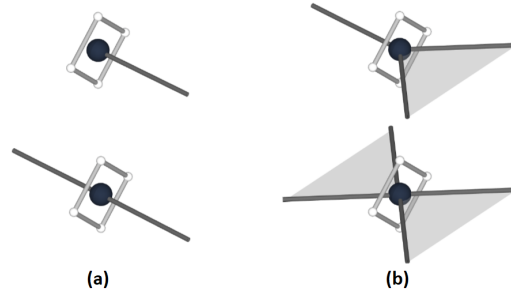

Figure 10: *Connection surfaces on joints, BL-joints and LL-joints are composed of two quads (one hemisphere per incident branch or leaflet). A single quad is used on extremities. The geometry of these connection surfaces is unconstrained on joints and extremities at this stage (a), whereas it is given by the normal of the incident leaflets in BL-joints and LL-joints (b).*

Details about how the hexahedra created around leaflets are registered within these connection surface, in a way that is fully compatible with the previous case, are given in section 3.5.

Connection surfaces are also constructed on joints, BL-joints and LL-joints (Figure 10). They are composed of two quads, each covering a hemisphere of the partitioned sphere. In the case of simple joints, even if there is no particular challenge in connecting two consecutive blocks of four hexahedra coming from both sides of a joint, these connection surfaces allow the upcoming hex sewing step to be generic and able to run in parallel. In the same spirit, a single quad is created on each extremity of the skeleton. In the BL-joints and LL-joints cases, the connection surface will allow hexahedra created around leaflets to be managed in the exact same way as in the branching points case.

The geometry of these connection surfaces is not fully constrained in the cases of joints and extremities at this stage of the algorithm. Indeed, the only available geometric information is the normal plane of the branch at these vertices. The rotation degree of freedom around the tangent direction will be set in the next step. In contrast, the geometry of the connection surfaces on BL-joints and LL-joints is set using the normal of the incident leaflets.

Around fan edges, an order among the incident faces is computed and also encoded in connection surfaces, as illustrated in Figure 11. On each fan edge, after sorting the incident faces on the plane orthogonal to the edge, a polyhedron composed of one two-sided face per incident face is constructed. Hexahedra that will be built on both sides of the leaflet faces will be registered within the edges of this connection surface allowing pairs of hexahedra to be sewn together independently afterwards. Unlike the quad connection sur-

faces used on branching points and joints, this connection surface only carries connectivity information while all the necessary geometric information is carried by the underlying meso-skeleton vertices.
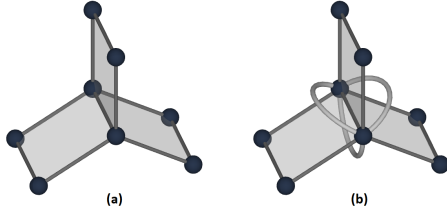

Figure 11: *After ordering the incident faces in the plane orthogonal to the edge, a connection surface is built around a fan edge. It is composed of a two-sided face per incident face.*

The construction of all these connection surfaces is independent. All the sphere partitioning and faces ordering computations can be processed in parallel.

3.4. Geometry propagation

As stated above, the geometry of the connection surfaces on joints and extremities is not fully set yet. However, the geometry of the connection surfaces on branching points and BL-joints is fixed. To find the positions that will generate the minimal torsion in the final hexahedra, the geometry of these connection surfaces is propagated using rotation minimizing frames (RMF) [PRW*18, VKB21].

First, frames are built on the connection surfaces of branching points and BL-joints. For each incident branch, the diagonals of the corresponding quad and the branch direction are used to initialize a frame. RMF is then used to propagate these frames along the branches, joint after joint. For branches that have a constrained frame on both ends, the propagation is first performed in one direction. The angular defect between the propagated frame and the frame of the quad on the other end of the branch is then distributed evenly among the joints using the arc-length parameterization along the branch in a backward step. The frame of each joint is finally used to set the geometry of their connection surface vertices (Figure 12).

Each branch of the meso-skeleton can be processed independently and the propagation of frames on all branches can be run in parallel.
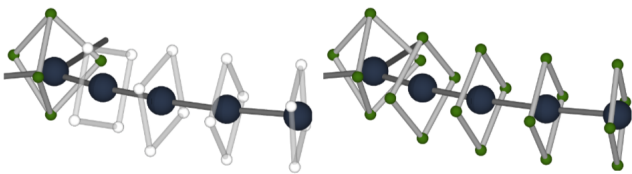

Figure 12: *The constrained geometry of the connection surfaces on branching points is propagated along the branches using Rotation Minimizing Frames.*

3.5. Blocks insertion

With the connection surfaces constructed, the domain is ready to be decomposed and filled with different types of construction blocks. [VKB21] defined a single generic block to reconstruct tubular domains: the *chunk*. In order to handle leaflets, we define two new types of blocks: *plates* and *half-chunks*. On each branch edge, a *chunk* composed of four hexahedra is inserted (Figure 13 (a)). The hexahedra are registered within the edges of the corresponding quads of the connection surfaces present on both sides of the edge. For each face of the leaflets, a *plate*, i.e. a pair of hexahedra, is inserted (Figure 13 (b)) and registered within the edges of the face. If one of the edges of the face is a fan edge, the hexahedra are registered within the edges of the corresponding two-sided face of the connection surface of the fan edge (Figure 13 (c)). On each boundary edge of the leaflets, a *half-chunk*, i.e. a pair of hexahedra, is inserted and registered within that boundary edge (Figure 13 (d)).
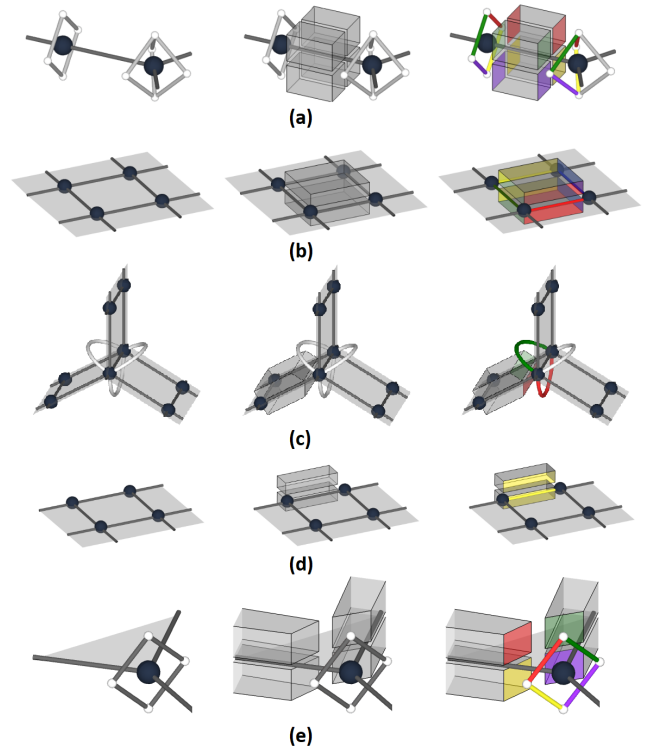

Figure 13: *Construction blocks are inserted and registered. (a) A chunk of four hexahedra on a curve edge is registered within the corresponding quads of its incident vertices connection surfaces. (b) A pair of hexahedra on a leaflet face is registered within its incident edges. (c) Around fan edges, these hexahedra are also registered within the two-sided faces of the connection surface. (d) A pair of hexahedra is registered within each boundary edge of a leaflet. (e) When a boundary vertex of a leaflet is a branching point or joint (here, a BL-joint), the two pairs of hexahedra attached to the the two incident leaflet boundary edges are also registered within the corresponding quad of the connection surface of the branching point or joint.*

If the boundary edge is incident to a branching point, a BL-joint or a LL-joint, the two incoming *half-chunks* are registered within the edges of the corresponding quad of the connection surface (Figure 13 (e)). This way, the interface exposed by leaflets on branching points and joints is fully compatible with the one exposed by the four-hexahedra chunks built on the curve branches.

Once this step is completed, each edge of the different connection surfaces is associated with exactly two hexahedra. Interior edges of the leaflets are associated with the hexahedra that lie around its two incident faces. Boundary edges of the leaflets are associated with the hexahedra of its incident face and the ones that have been inserted on the boundary.

If the required place for each construction block has been allocated beforehand in the underlying data containers, the creation of all these hexahedra can also be run in parallel.

3.6. Volumes sewing

To complete the topology of the raw mesh, the generated hexahedra remain to be connected together. The connection surfaces and edges of the leaflets already carry all the information needed to establish these connections. As illustrated in Figure 14 (a) and (b), the pairs of hexahedra associated with the edges of the connection surfaces, either on branching points or around fan edges, are sewn together. Traversing the edges of all connection surfaces to perform these connections can be done in parallel.

For all interior and boundary leaflet edges, the pairs of hexahedra that lie on both sides of the edges are connected together (Figure 14 (c)). Again, traversing the edges of all leaflets to perform these connections can be done in parallel.
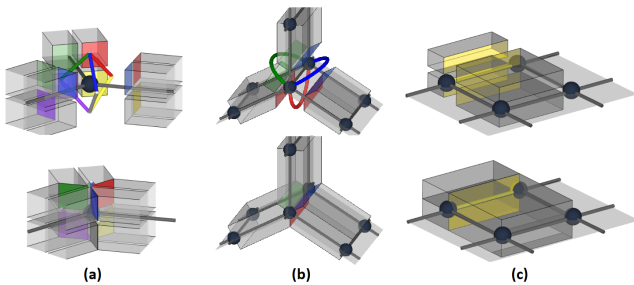

Figure 14: *Hexahedra built around the meso-skeleton curve and surface parts are sewn together. (a) Each edge of each connection surface is associated to a pair of hexahedra to connect. (b) Each edge of each leaflet is associated to two pairs of hexahedra to connect (over and under the leaflet surface).*

Despite what all the illustrations suggest so far, the obtained volume mesh still has no geometry, but its combinatorial topology is complete and has the same genus as the input domain.

3.7. Geometry transfer

All the necessary information to set the geometry is carried by the connection surfaces and the leaflets of the meso-skeleton.

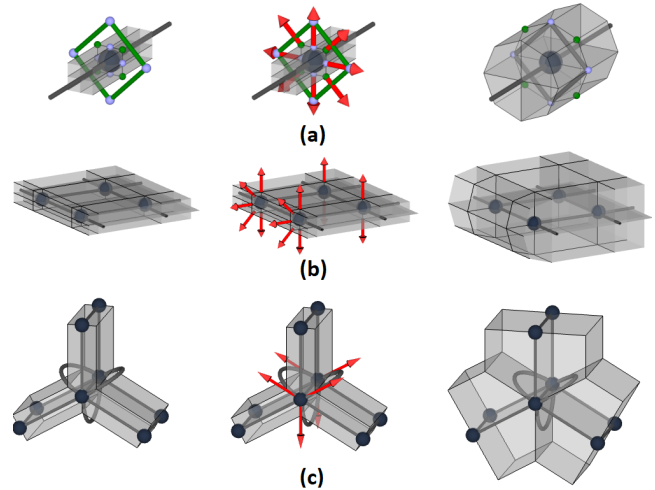Each connection surface is associated with several vertices in

Figure 15: *Transfer of geometric information from the skeleton to the hex mesh. (a) Transfer from a joint with a connection surface. (b) Transfer from a leaflet and its edge. (c) Transfer from a fan edge.*

the hexahedral mesh. The central vertex, which is placed at the position of the corresponding meso-skeleton vertex. One vertex for each vertex of the connection surface, that are placed at the position already computed for the vertices of the connection surface. One vertex per edge of the connection surface, that are placed halfway between the vertices of the edge, on the sphere of local radius. This is illustrated in Figure 15 (a)) for a joint vertex. The same process applies to any connection surface.

Each interior vertex of the leaflets is associated with three vertices of the hexahedral mesh. The central vertex is placed at the leaflet vertex position. The other two are pushed up and down along the normal vector, using the local radius (Figure 15 (b)).

Leaflet vertices that lie on the boundary (except branching points and joints), have three additional vertices to position. The central one is pushed in the plane of the leaflet towards the exterior. The other two are pushed towards an average between this direction and the up and down normals of the leaflet (Figure 15 (b)).

Finally, leaflet vertices that lie on fan edges are associated to as many vertices as the degree of the edge, plus one central vertex. The central vertex is placed at the meso-skeleton vertex position. The other vertices are pushed outwards, again using the local radius, in a direction that is halfway between that of the pairs of successive incident faces (Figure 15 (c)).

Each branching point, joint and leaflet vertex of the meso-skeleton can be processed independently. For this step again, the algorithm can be run in parallel. After that, the raw hexahedral mesh is complete.

4. Results

The described pipeline has been implemented in C++ using an incidence graph structure for the meso-skeleton, a 2-dimensional combinatorial map for the connection surfaces and a 3-dimensional

P. Viville & P. Kraemer & D. Bechmann / Meso-Skeleton Guided Hexahedral Mesh Design

combinatorial map for the hex mesh. The code and used models are publicly available [CGo].

This extended algorithm can handle skeletons with many combinations of edges and faces, ranging from an edge-only skeleton to a face-only skeleton with or without border, as shown in Figure 16 with a bronchial system and a hollow torus.
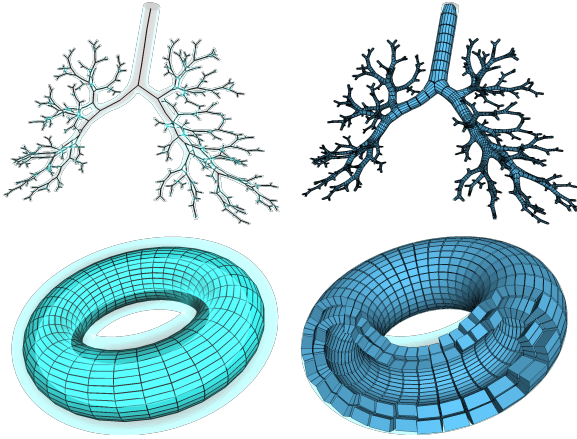

Figure 16: *Example of application to a purely curve skeleton and a purely surface skeleton with no borders.*

It results in a raw mesh (Figure 17), a decomposition of the input domain into hexahedral blocks that shares the topological characteristics of the domain with a crude approximation of its geometry and a minimal amount of irregular edges. To be usable for example in a finite elements simulation application context, this mesh can be further refined and its geometry optimized to better fit the domain. In other contexts like isogeometric analysis in which higher order elements are used and less hexahedra are needed, the raw mesh may not be topologically further refined [HCB05].
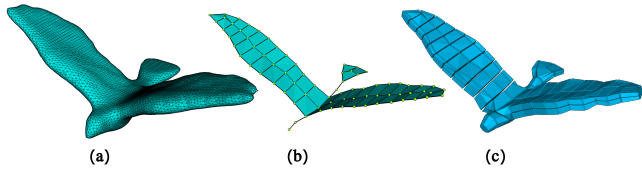

Figure 17: *Example of application to the Eagle model using a meso-skeleton. (a) The initial domain, (b) The meso-skeleton, (c) The generated raw hexahedral mesh.*

The targeted resolution of the mesh depends on the needs of the application context. In many cases, the mesh should fit the surface as closely as possible while ensuring that all hexahedra have a shape as close as possible to that of a rectangular cuboid. To achieve this, existing methods can be used to optimize both connectivity and geometry. Primal subdivision increases the resolution of the mesh. Padding adds a layer of hexahedra on the surface of the whole mesh. As a result, each boundary hexahedron only exposes one face to the boundary, allowing for better angles optimization. The structure of the generated mesh and the orientations provided by the underlying meso-skeleton also allow to control the subdivisions to be made in either longitudinal or radial directions along branches. Several geometry optimization methods exist [LSVT15] that generally consists in a local-global optimization loop. Local steps compute ideal directions and lengths for each edge based on its neighbourhood. Global steps solve a global problem to find a least squares solution that satisfies those quantities while enforcing boundary vertices to stay on the domain surface. As a result, vertices of the mesh are moved iteratively, resulting in a mesh in which angles tend towards orthogonality and therefore bring hexahedra closer to cuboids.

Several measures can denote the quality of a hexahedral mesh, the most prevalent one being the scaled Jacobian, computed for each element of the mesh. A negative value indicates an inverted element, which generally discards the mesh as a usable support for any computation, and rectangular cuboids have a value of 1. The worst and the average value over the whole mesh are usually given as good indicators of the global quality of a mesh.

Figure 18 shows several meshes obtained using our method. For each model, the silhouette of the shape is shown along with its associated meso-skeleton next to the resulting hexahedral mesh. Table 1 indicates the number of hex, minimum and average scaled Jacobian for each model. Some of the presented models were already processed by previous works and their associated data is presented in the table. The surface for Chair was taken from the Aim@Shape Repository. The surfaces and skeletons of DualOcta, TorusTwist, Twist, and VerticalTurbine were modeled to provide some complex cases. TorusTwist for example is the intersection of two Moebius strips. In most cases, our method is able to produce meshes that have better quality elements.

Our hex mesh construction process generates a limited and predictable amount of irregularities within the raw mesh that can only appear under a few conditions. Figure 19 illustrates the singular structure of the models shown in Figure 18. Irregular edges are present around each branching point of the skeleton, depending on

(#) Model	Previous works			Our results	
	[ref]	Min/Avg SJ	#Hex	Min/Avg SJ	#Hex
(1) Chair				.262/.941	12128
(2) Dilo	[LZS*21]	**.551**/.949	36500	.380/**.968**	39232
(3) DualOcta				.657/.873	1344
(4) Eagle	[GSP19]	.334/.821	18306	**.633**/**.948**	7072
(5) Hand	[LPP*20]	.330/.721	1300	**.629**/**.918**	3584
(6) Holes	[LPP*20]	.104/.746	858	**.725**/**.938**	2432
(7) Knob	[LPP*20]	.555/1.00	744	**.900**/1.00	6912
(8) Metatron	[LPP*20]	.820/**.981**	6624	**.844**/.980	7008
(9) Pinion	[LPP*20]	**.966**/**.986**	240	.836/.973	16640
(10) Plane	[GSP19]	.265/.805	54267	**.501**/**.942**	6496
(11) Sculpture	[BGMC22]	.689/961	23936	**.836**/**.984**	2240
(12) Torque	[LPP*20]	.576/.858	880	**.616**/**.905**	1144
(13) TorusTwist				.690/.878	768
(14) Twist				.802/.968	4224
(15) Ujoint	[LPP*20]	**.683**/.865	112	.640/**.991**	7232
(16) VertTurbine				.506/.957	15168

Table 1: *Minimum and average scaled Jacobian of the generated meshes including those presented in Figure 18.*

Figure 18: *Several meshes obtained using our algorithm with post processing using padding, simple subdivision, and geometric optimization. For each model, the left image shows the silhouette of the domain and the meso-skeleton and the right image shows the obtained hex mesh. Quality measures for these meshes are indicated in Table 1.*
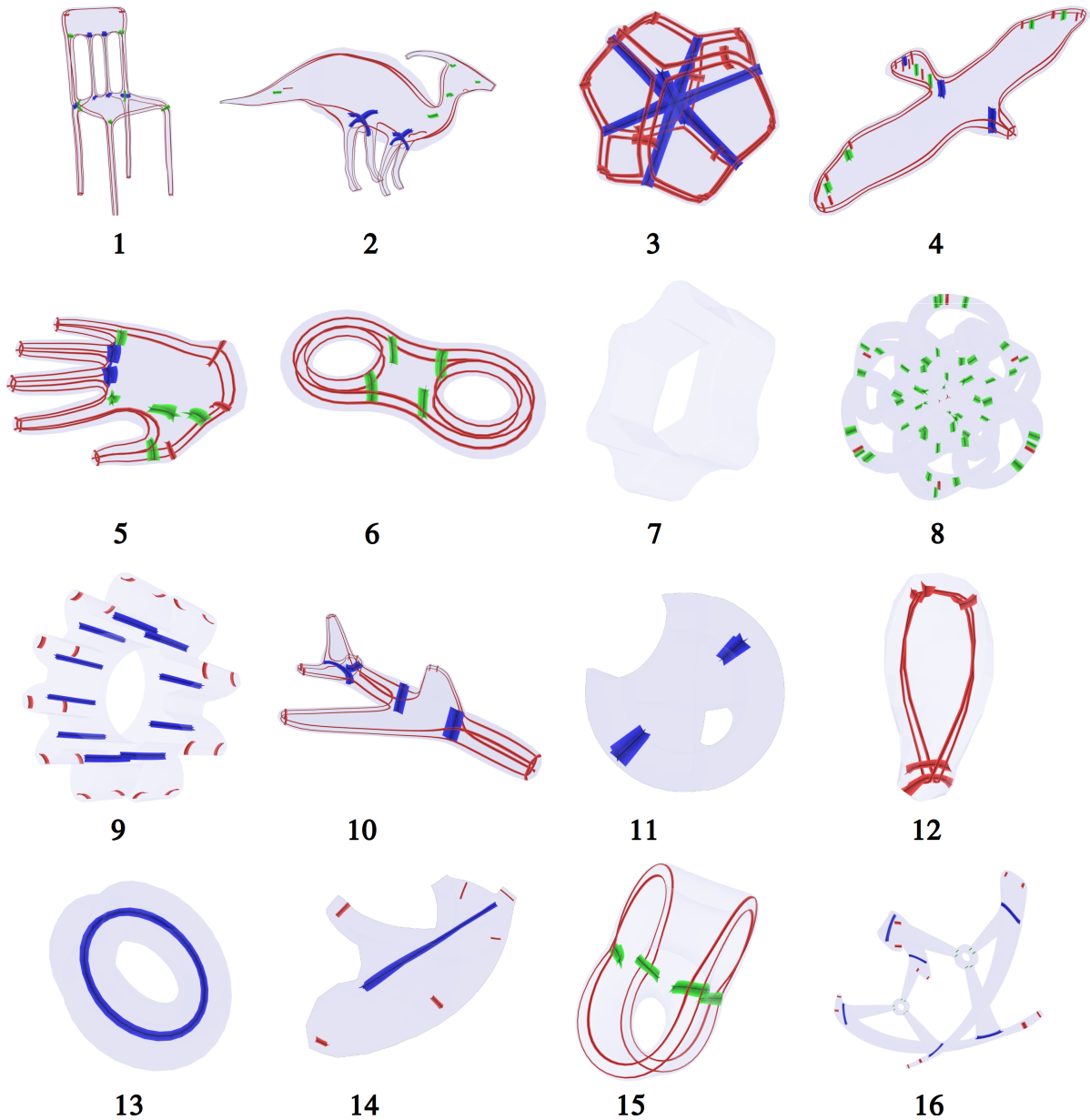
Figure 19: *Singular structures of the meshes presented in Figure 18. In red: edges of degree 3. In green: edges of degree 5. In blue: edges of degrees 6 and above. These images have been produced in Hexalab [BTP*19].*

its degree. Two irregular edges are created around each BL-joint, LL-joint and irregular vertex of leaflets, one above, and one below (see for example the irregular edges on the knuckles of the Hand, or the head and tail of the Eagle). One irregular edge is present for each fan edge of the skeleton. The addition of a padding layer during post processing creates additional irregularities: four irregular edges of degree 3 along each branch edge and two irregular edges of degree 3 along each leaflet boundary edge. These create loops that are clearly visible in red in Figure 19. Irregularities around branching points can however be avoided in some configurations. [VKB21] proposes to insert a compatible 8-hex chunk in

branching points of degree 6 or less whose incident branches are mutually almost aligned or orthogonal. In this case, no irregular edge is created around these branching points. In the same fashion, a compatible 4-hex chunk could be inserted in fan edges of degree 4 or less whose incident faces are mutually almost aligned or orthogonal, also resulting in the creation of no irregular edge around these fan edges.

Figure 20 shows side-to-side the singular structures of meshes obtained using our approach and using some other existing methods. As the singular graph is determined by the skeleton rather
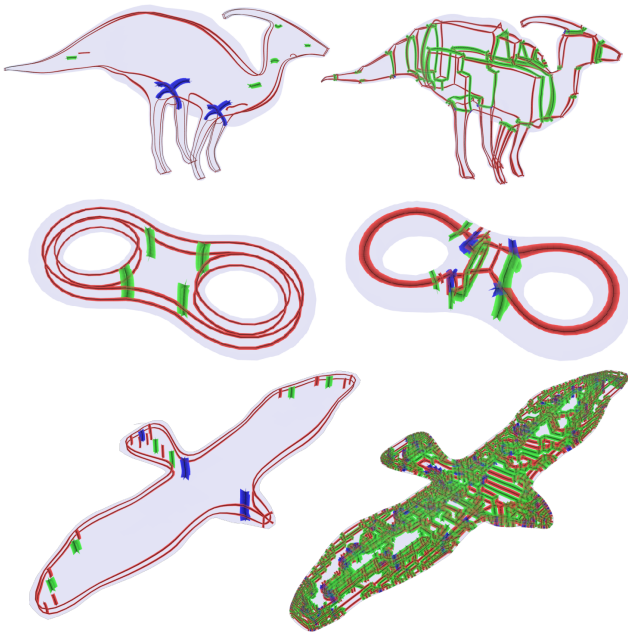
Figure 20: *Singular structure of several meshes. On the left, our results. On the right, results obtained by different methods: Dilo [LZS*21], Holes [LPP*20], and Eagle [GSP19].*

| (#) Model | Our results | | | Prior works |
	Skeleton	Raw	Optim	times
(1) Chair	HD 5min	**7ms**	45s	-
(2) Dilo	MC 5min	**3ms**	90s	15min [LZS*21]
(3) DualOcta	P	**16ms**	60s	-
(4) Eagle	HD 10min	**6ms**	45s	N/A [GSP19]
(5) Hand	MC 3min	**3ms**	45s	2.3min [LPP*20]
(6) Holes	MC 1min	**2ms**	30s	6.5min [LPP*20]
(7) Knob	HD 5min	**4ms**	60s	1min [LPP*20]
(8) Metatron	MC 10min	**14ms**	45s	2min [LPP*20]
(9) Pinion	HD 5min	**3ms**	60s	1min [LPP*20]
(10) Plane	HD 5min	**4ms**	45s	N/A [GSP19]
(11) Sculpture	MC 10min	**4ms**	80s	N/A [BGMC22]
(12) Torque	HD 3min	**2ms**	30s	0.6min [LPP*20]
(13) TorusTwist	P	**6ms**	45s	-
(14) Twist	P	**7ms**	45s	-
(15) Ujoint	HD 3min	**2ms**	45s	0.1min [LPP*20]
(16) VertTurbine	P	**3ms**	60s	-

Table 2: *Time required to produce the results presented in Figure 18. For our results, the given times are split into three distinct tasks: produce the skeleton (**P**: Procedural generation, **MC**: edited Mean Curvature skeleton, **HD**: Hand Drawn), generate the raw hexahedral mesh, subdivide and optimize the geometry.*

than computed, our method generally produces more predictable irregularities, especially compared to grid based methods.

General purpose hexahedral mesh generation methods usually involve costly computations and their execution times are generally expressed in the order of several minutes. Our execution times are presented in Table 2. They are separated in three distinct steps: skeleton production, raw hexahedral mesh generation, and optimization of the mesh.

In our current implementation, skeleton production and optimization steps are user controlled. Therefore the times given are rough measurements of the time needed by an experienced user. Raw hexahedral mesh generation times are precise measurements. This step only takes a few milliseconds even for skeletons composed of hundreds of branching points.

The skeletons used for our results were produced in three way: procedural generation along with their corresponding surface, hand modeled, or edited starting from a computed mean curvature skeleton [TAOZ12]. The process of editing or drawing a skeleton was generally a matter of a few minutes. When applied, padding and subdivision steps require milliseconds. The subsequent geometry optimization is a more costly process, especially as the mesh size increases. A few hundred iterations of the geometry optimization loop are usually required. However, given a fixed connectivity, the least square matrix involved in the process only has to be factorized once, saving much time for the subsequent iterations. The needed time varies depending on model size (per 100 iterations: 700ms for 7000 hex; 13s for 56000 hex).

With an improved editor that proposes a first guess using an automatic curve skeleton or meso-skeleton extraction method, we expect uninitiated users to be able to produce such skeletons in a similar time frame, especially since given the post-processing optimizations, geometric accuracy is not required on the skeleton itself. Furthermore, letting the user intervene in the process may lead to a more compliant result. As the production of the raw mesh and minor optimizations could be done in near real time during edition, an efficient interactive loop between skeleton edition and hex-mesh generation would remain rather efficient.

5. Limitations

Although our algorithm can manage a wide variety of meso-skeleton configurations, some specific cases are still not handled by the currently defined construction blocks and assembly process. Among the three identified configurations, two are actually compatible with our pipeline but are not yet handled in our implementation, and one is incompatible with our block structure and would require a local modification of the skeleton to be processed.

The first configuration is the case of a branch incident to an internal vertex of a leaflet, as illustrated in Figure 21. The interface exposed by the incoming branch cannot always find a compatible docking interface. However, a modification of the leaflet mesh around the vertex would allow the insertion of a connection block on which the branch could connect. The downside is the introduction of new irregular vertices in the leaflet connectivity.

The second configuration concerns fan edges stopping on an internal vertex of a leaflet, as shown in Figure 22. The hexahedra inserted on the boundary of the incoming leaflet cannot be connected around this vertex. Here again, a modification of the local connectivity and the insertion of an interface block can correct the problem, at the price of some new irregular vertices.
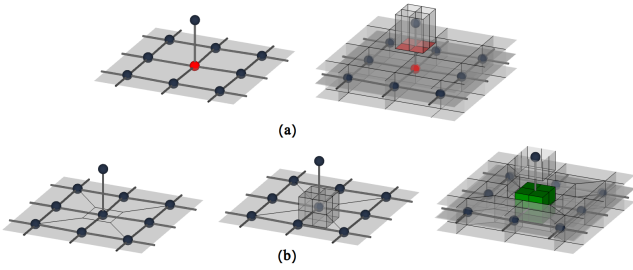
Figure 21: *Branches cannot be incident to a leaflet internal vertex. (a) With the generic construction process, the interface exposed by the incoming branch (in red) cannot be connected on the leaflet. (b) A modification of the leaflet connectivity around the vertex is required (bottom left) and the insertion of an interface block leads to a compatible configuration.*
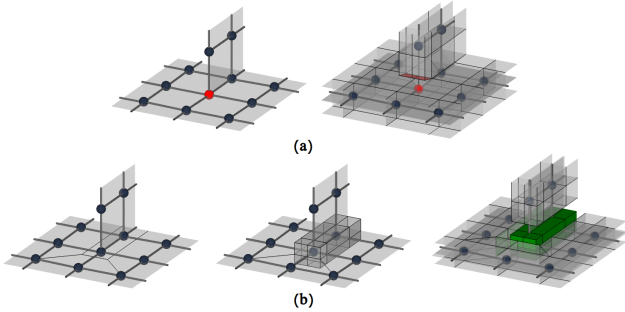

Figure 22: *Fan edges cannot stop on an internal vertex of a leaflet. (a) With the generic construction process, the hexahedra of the boundary of the incoming second leaflet (in red) cannot be connected on the first leaflet. (b) A modification of the connectivity around the vertex in the second leaflet is required (bottom left) and the insertion of an interface block leads to a compatible configuration.*

The third configuration occurs when a branch is incident to a boundary vertex of a fan edge, as illustrated in Figure 23. The interface exposed on this vertex by the hexahedra inserted along the leaflet boundary edges is not compatible with the interface exposed by the incident branch. This case requires a local modification of the connectivity to bring the skeleton back to a compatible configuration: either move the branch away from the fan edge or make the fan edge stop before the boundary.

Geometric difficulties may also arise at joints or branching points where the incident branches or leaflets directions are not sufficiently distributed on the sphere (Figure 24). These kind of configuration can lead to spiky hexahedra with small angles that are hard to improve. A solution could be to insert virtual branches in the branching points to improve the distribution of the directions. A single chunk of 4 hexahedra would be added for these virtual branches and the geometry of their extremity would lie on the sphere of local radius.
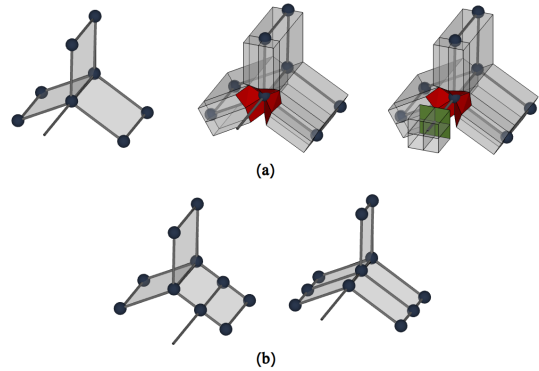

Figure 23: *Branches cannot be incident to a boundary vertex of a fan edge. (a) In this configuration, the hexahedra inserted on the boundary edges of the leaflets expose an interface composed of more than 4 hexahedra (in red) that is not compatible with the 4 hexahedra that are exposed by the incoming incident branch (in green). (b) Such configurations need to be eliminated by either moving the branch away from the fan edge (bottom left), or retracting the fan edge (bottom right).*
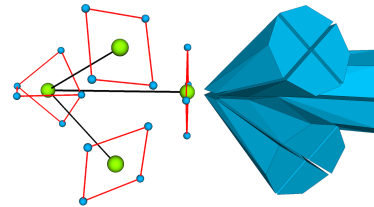

Figure 24: *Badly distributed branch directions can lead to thin, hard to improve, angles in the resulting hexahedra. Virtual branches could help mitigate this problem.*

6. Conclusion

In this article, we proposed a new hexahedral mesh generation pipeline. It starts from a surface of the domain to mesh and its meso-skeleton, composed of both curve and surface parts. This proxy of the domain shape serves as a support for the construction of a raw hexahedral mesh. Connection surfaces are built on the different types of branching points and fan edges of the skeleton to encode how the hexahedral blocks that meet at these points should be connected. The locality and independence of most steps allow each of them to be run in parallel, leading to a very efficient implementation. The meso-skeleton naturally follows the geometry of the domain, and so does the resulting mesh. The basic building blocks used to build the mesh tend to a mostly regular connectivity, irregularities being mainly localized around branching points and irregularities in the leaflets. Compared to previous methods based on curve-only skeletons, thanks to its support of surface parts in the skeleton, our approach is able to address a much wider range of geometric domains.

Several possibilities could be explored in order to improve this work. As of yet, our pipeline can be automated once a suitable meso-skeleton is provided. Progress on the development of reliable

meso-skeleton extraction methods combined with off the shelf surface remeshing algorithms will allow the completion of a fully automated pipeline with minimal user input required. User could still stay in control of the result in an efficient interactive loop of skeleton edition and hex-mesh generation. Also, automatic repairing or transformation of the meso-skeleton could enable the support of the currently incompatible configurations, which could further expand the range of possible shapes to be processed. Finally, the proposed evolution using virtual branches could allow to generate meshes of better quality in some challenging geometric configurations.

References

[BGMC22] BRÜCKLER H., GUPTA O., MANDAD M., CAMPEN M.: The 3d motorcycle complex for structured volume decomposition. *Computer Graphics Forum 41*, 2 (2022), 221–235. doi:https://doi.org/10.1111/cgf.14470. 1, 8, 11

[Bla00] BLACKER T.: Meeting the challenge for automated conformal hexahedral meshing. *Proceedings of the 9th International Meshing Roundtable* (11 2000). 1

[BRK*22] BEAUFORT P.-A., REBEROL M., KALMYKOV D., LIU H., LEDOUX F., BOMMES D.: Hex me if you can. *Computer Graphics Forum 41*, 5 (2022), 125–134. doi:https://doi.org/10.1111/cgf.14608. 1

[BTP*19] BRACCI M., TARINI M., PIETRONI N., LIVESU M., CIGNONI P.: Hexalab.net: An online viewer for hexahedral meshes. *Computer-Aided Design 110* (2019), 24 – 36. doi:https://doi.org/10.1016/j.cad.2018.12.003. 10, 13

[CGo] CGOGN: ICube, IGG group. URL: https://github.com/cgogn/CGoGN_3. 8

[DFOL22] DECROOCQ M., FRINDEL C., OHTA M., LAVOUÉ G.: Modeling and hexahedral meshing of arterial networks from centerlines. *ArXiv abs/2201.08279* (2022). 2

[DLX*22] DOU Z., LIN C., XU R., YANG L., XIN S., KOMURA T., WANG W.: Coverage axis: Inner point selection for 3d shape skeletonization. *Computer Graphics Forum 41*, 2 (2022), 419–432. doi:https://doi.org/10.1111/cgf.14484. 3, 4

[DSH20] DOCAMPO SÁNCHEZ J., HAIMES R.: A regularization approach for automatic quad mesh generation. In *Proceedings of the 28th International Meshing Roundtable* (10 2020). 4

[GSP19] GAO X., SHEN H., PANOZZO D.: Feature preserving octree-based hexahedral meshing. *Computer Graphics Forum 38* (08 2019), 135–149. doi:10.1111/cgf.13795. 1, 8, 11

[HCB05] HUGHES T., COTTRELL J., BAZILEVS Y.: Isogeometric analysis: Cad, finite elements, nurbs, exact geometry and mesh refinement. *Computer Methods in Applied Mechanics and Engineering 194*, 39 (2005), 4135–4195. URL: https://www.sciencedirect.com/science/article/pii/S0045782504005171, doi:https://doi.org/10.1016/j.cma.2004.10.008. 8

[KBLK14] KREMER M., BOMMES D., LIM I., KOBBELT L.: Advanced automatic hexahedral mesh generation from surface quad meshes. In *Proceedings of the 22nd International Meshing Roundtable* (2014), pp. 147–164. 1

[KLF16] KOWALSKI N., LEDOUX F., FREY P.: Smoothness driven frame field generation for hexahedral meshing. *Computer-Aided Design 72* (2016), 65 – 77. 23rd International Meshing Roundtable Special Issue: Advances in Mesh Generation. doi:https://doi.org/10.1016/j.cad.2015.06.009. 1

[LCLJ10] LIU L., CHAMBERS E. W., LETSCHER D., JU T.: A simple and robust thinning algorithm on cell complexes. *Computer Graphics Forum 29*, 7 (2010), 2253–2260. doi:https://doi.org/10.1111/j.1467-8659.2010.01814.x. 3

[LMPS16] LIVESU M., MUNTONI A., PUPPO E., SCATENI R.: Skeleton-driven adaptive hexahedral meshing of tubular shapes. *Computer Graphics Forum 35*, 7 (2016), 237–246. 2, 4

[LPC21] LIVESU M., PITZALIS L., CHERCHI G.: Optimal dual schemes for adaptive grid based hexmeshing. *ACM Transactions on Graphics 41*, 2 (dec 2021). doi:10.1145/3494456. 1

[LPP*20] LIVESU M., PIETRONI N., PUPPO E., SHEFFER A., CIGNONI P.: Loopycuts: Practical feature-preserving block decomposition for strongly hex-dominant meshing. *ACM Transactions on Graphics 39*, 4 (2020). doi:(10.1145/3386569.3392472). 1, 8, 11

[LQS17] LU J. H.-C., QUADROS W. R., SHIMADA K.: Evaluation of user-guided semi-automatic decomposition tool for hexahedral mesh generation. *Journal of Computational Design and Engineering 4*, 4 (05 2017), 330–338. doi:10.1016/j.jcde.2017.05.001. 1

[LSVT15] LIVESU M., SHEFFER A., VINING N., TARINI M.: Practical hex-mesh optimization via edge-cone rectification. *ACM Transactions on Graphics 34*, 4 (2015). doi:10.1145/2766905. 8

[LVS*13] LIVESU M., VINING N., SHEFFER A., GREGSON J., SCATENI R.: Polycut: Monotone graph-cuts for polycube base-complex construction. *ACM Transactions on Graphics 32*, 6 (nov 2013). URL: https://doi.org/10.1145/2508363.2508388, doi:10.1145/2508363.2508388. 1

[LWS*16] LI P., WANG B., SUN F., GUO X., ZHANG C., WANG W.: Q-mat: Computing medial axis transform by quadratic error minimization. *ACM Transactions on Graphics 35*, 1 (dec 2016). doi:10.1145/2753755. 3, 4

[LZJ*17] LEI N., ZHENG X., JIANG J., LIN Y.-Y., GU D. X.: Quadrilateral and hexahedral mesh generation based on surface foliation theory. *Computer Methods in Applied Mechanics and Engineering 316* (2017), 758–781. Special Issue on Isogeometric Analysis: Progress and Challenges. URL: https://www.sciencedirect.com/science/article/pii/S0045782516312762, doi:https://doi.org/10.1016/j.cma.2016.09.044. 1

[LZS*21] LI L., ZHANG P., SMIRNOV D., ABULNAGA S. M., SOLOMON J.: Interactive all-hex meshing via cuboid decomposition. *ACM Transacions on Graphics 40*, 6 (dec 2021). doi:10.1145/3478513.3480568. 1, 8, 11

[NRP11] NIESER M., REITEBUCH U., POLTHIER K.: Cubecover-parameterization of 3d volumes. *Computer Graphics Forum 30*, 5 (2011), 1397–1406. URL: https://doi.org/10.1111/j.1467-8659.2011.02014.x, doi:10.1111/j.1467-8659.2011.02014.x. 1

[OS16] OLIVEIRA B., SUNDNES J.: Comparison of tetrahedral and hexahedral meshes for finite element simulation of cardiac electro-mechanics. In *VII European Congress on Computational Methods in Applied Sciences and Engineering* (01 2016), pp. 164–177. doi:10.7712/100016.1801.9193. 1

[OSCS99] OWEN S. J., STATEN M. L., CANANN S. A., SAIGAL S.: Q-morph: an indirect approach to advancing front quad meshing. *International journal for numerical methods in engineering 44*, 9 (1999), 1317–1340. 4

[PAR*20] PAPADIMITRAKIS D., ARMSTRONG C., ROBINSON T., MOIGNE A., SHAHPAR S.: Building direction fields on the medial object to generate 3d domain decomposaitions for hexahedral meshing. In *Proceedings of 28th International Meshing Roundtable* (02 2020). doi:10.5281/zenodo.3653428. 2

[PCS*22] PIETRONI N., CAMPEN M., SHEFFER A., CHERCHI G., BOMMES D., GAO X., SCATENI R., LEDOUX F., REMACLE J., LIVESU M.: Hex-mesh generation and processing: A survey. *ACM Transactions on Graphics 42*, 2 (oct 2022). doi:10.1145/3554920. 1

[PMA21] PELTIER S., MORIN G., AHOLOU D.: Tubular parametric volume objects: Thickening a piecewise smooth 3d stick figure. *Computer Aided Geometric Design 85* (2021), 101981. doi:https://doi.org/10.1016/j.cagd.2021.101981. 2

[PRW*18] PANOTOPOULOU A., ROSS E., WELKER K., HUBERT E., MORIN G.: Scaffolding a skeleton. *Research in Shape Analysis 12* (2018), 17–35. 6

[Qua16] QUADROS W. R.: Laytracks3d: A new approach for meshing general solids using medial axis transform. *Computer-Aided Design 72* (2016), 102–117. 23rd International Meshing Roundtable Special Issue: Advances in Mesh Generation. doi:https://doi.org/10.1016/j.cad.2015.08.002. 2

[RLS*12] REMACLE J.-F., LAMBRECHTS J., SENY B., MARCHANDISE E., JOHNEN A., GEUZAINET C.: Blossom-quad: A non-uniform quadrilateral mesh generator using a minimum-cost perfect-matching algorithm. *International journal for numerical methods in engineering 89*, 9 (2012), 1102–1119. 4

[Sch16] SCHNEIDERS R.: A grid-based algorithm for the generation of hexahedral element meshes. *Engineering with Computers 12*, 3 (2016), 168–177. 1

[SRUL16] SOKOLOV D., RAY N., UNTEREINER L., LÉVY B.: Hexahedral-dominant meshing. *ACM Transactions on Graphics 35*, 5 (2016), 1–23. doi:https://doi.org/10.1145/2930662. 1

[Tak19] TAKAYAMA K.: Dual sheet meshing: An interactive approach to robust hexahedralization. *Computer Graphics Forum 38*, 2 (2019), 37–48. doi:https://doi.org/10.1111/cgf.13617. 1

[TAOZ12] TAGLIASACCHI A., ALHASHIM I., OLSON M., ZHANG H.: Mean curvature skeletons. *Computer Graphics Forum 31*, 5 (2012), 1735–1744. doi:https://doi.org/10.1111/j.1467-8659.2012.03178.x. 3, 4, 11

[TDS*16] TAGLIASACCHI A., DELAME T., SPAGNUOLO M., AMENTA N., TELEA A.: 3d skeletons: A state-of-the-art report. *Computer Graphics Forum 35*, 2 (2016), 573–597. doi:https://doi.org/10.1111/cgf.12865. 2

[ULP*15] USAI F., LIVESU M., PUPPO E., TARINI M., SCATENI R.: Extraction of the quad layout of a triangle mesh guided by its curve skeleton. *ACM Transactions on Graphics 35*, 1 (2015), 6:1–6:13. doi:10.1145/2809785. 2

[VKB21] VIVILLE P., KRAEMER P., BECHMANN D.: Hexahedral mesh generation for tubular shapes using skeletons and connection surfaces. In *Proceedings of the 16th International Joint Conference on Computer Vision, Imaging and Computer Graphics Theory and Applications - GRAPP* (2021), pp. 45–54. doi:10.5220/0010222000450054. 2, 3, 4, 5, 6, 10

[VPR19] VERHETSEL K., PELLERIN J., REMACLE J.-F.: Finding hexahedrizations for small quadrangulations of the sphere. *ACM Transactions on Graphics 38*, 4 (jul 2019). doi:10.1145/3306346.3323017. 5

[YSC*16] YAN Y., SYKES K., CHAMBERS E., LETSCHER D., JU T.: Erosion thickness on medial axes of 3d shapes. *ACM Transacions on Graph.ics 35*, 4 (jul 2016). doi:10.1145/2897824.2925938. 3