Pacific Graphics 2016
E. Grinspun, B. Bickel, and Y. Dobashi
(Guest Editors)

Volume 35 (2016), Number 7

Skeleton-driven Adaptive Hexahedral Meshing of Tubular Shapes

Marco Livesu
CNR IMATI, Genoa, Italy

Alessandro Muntoni
Università di Cagliari, Italy

Enrico Puppo
Università di Genova, Italy

Riccardo Scateni
Università di Cagliari, Italy

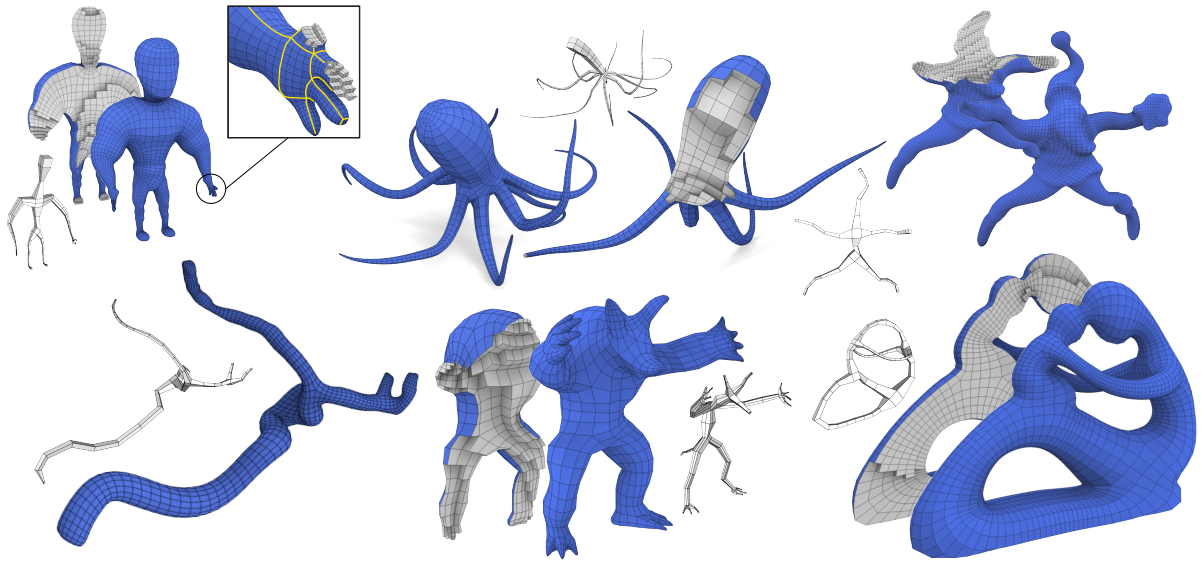


Figure 1: *A gallery of hexahedral meshes generated with our method without resolution scaling; the tubular structures used to derive the hexahedral meshes are shown in white. From top left to bottom right:* BIGBUDDY, OCTOPUS, SANTA, BLOODVESSEL, ARMADILLO *and* FERTILITY.

Abstract

We propose a novel method for the automatic generation of structured hexahedral meshes of articulated 3D shapes. We recast the complex problem of generating the connectivity of a hexahedral mesh of a general shape into the simpler problem of generating the connectivity of a tubular structure derived from its curve-skeleton. We also provide volumetric subdivision schemes to nicely adapt the topology of the mesh to the local thickness of tubes, while regularizing per-element size. Our method is fast, one-click, easy to reproduce, and it generates structured meshes that better align to the branching structure of the input shape if compared to previous methods for hexa mesh generation.

Categories and Subject Descriptors (according to ACM CCS): I.3.5 [Computer Graphics]: Computational Geometry and Object Modeling—Physically based modeling

1. Introduction

The quest for volumetric meshes for physically based simulations has dramatically increased in recent years. While classical applications deal with mechanical objects (e.g., in the aeronautical, mechanical and medical industries) new applications have emerged (e.g., in the movie and gaming industry, as well as in the biomedical field) which deal with natural shapes, often coming in the form of articulated objects. Special effects involving fluids and deformable objects are ubiquitous in many of the most recent movie productions and videogames. Likewise, the accurate simulation of human tissues

and organs widely extends the diagnostic power of CT data as well as of virtual surgery. No matter whether models are taken from reality or imagination, they are requested to behave and interact with the virtual world as if they were real.

Physically-plausible simulations [NMK*06] require a volumetric discretization of all elements of a scene interacting with each other. This need raises the bar for meshing algorithms, often shaped around precise target shapes [Bla01] and now required to provide new tools to automatically generate industry-ready meshes from shapes that are definitely different: no more mechanical parts exposing sharp edges and regular patterns, but shapes mimicking living beings.

To this aim, although high quality tetrahedral meshes can be reliably generated with existing methods [ACSYD05], hexahedral meshes are usually preferred due to their superior numerical properties and ability to keep the resolution lower [KBLK14, Bla01]. Structured hexahedral meshes are often preferred to semi-structured or unstructured ones because they can be more efficiently stored using specialized data structures, yet because they are the models of choice for many simulations where a strict alignment of elements is required [GDC15, RG11]. A hexahedral mesh is *structured* when it is composed of a single regular volume, or it can be decomposed into a few sub-volumes, each one with the connectivity of a regular grid [Tau04].

On the one hand, manually creating high-quality structured hexahedral meshes is a laborious task that can take days of work. On the other hand, automatic meshing is a challenging open problem, with much work happening in recent years (see Section 2).

In this work we restrict our attention to articulated shapes whose general structure is well captured by a curve-skeleton [TDS*16, CSM07]. Shapes in this category include, but are not limited to: humans and animals; articulated (possibly imaginary) creatures; tree-like structures like vessels; and plants. This restrictive choice to shapes for which a curve-skeleton can be extracted is based on the fact that physically-based simulations on such shapes are common not only in medicine and biology but also in the animation industry, where most characters created by digital artists belong to this category.

We propose an automatic algorithm that, taking in input a surface mesh and its curve-skeleton, produces a structured hexahedral mesh covering the volume bounded by the input surface. Our method is fast, one-click, easy to reproduce and it does not require any parameter tuning by the user. The hexahedral meshes we produce have high quality and nicely align with the main features of the target surface, a key component for accurate simulations [Bla01].

The main contribution of our paper is twofold:

- We extend the work of [ULP*15] from surfaces to volumes, automatically generating hexahedral meshes that directly encode the structure of the input shape, given by its curve-skeleton (Section 3);
- We propose a sampling technique for the curve-skeleton to control density *along* the skeleton arcs, and a set of volumetric subdivision schemes to control density *across* the skeleton arcs. Our density control system nicely adapts to the local thickness of the shape, minimizing the resolution of the model and reducing the variance of the element sizes (Section 5).

2. Related work

The generation of high quality hex meshes filling a target surface has been object of research since decades. An exhaustive survey of the literature in the field is beyond the scope of this paper. Here, we just recap only the most relevant approaches, grouped by meshing technique. We refer the reader to [Bla01, Tau01] for further details on classical approaches.

Skeleton-based approaches are the most related to our work. When the meshing process is driven by curve-skeleton, the critical part is the discretization of junctions, where incoming arcs from different directions meet. In [LLD12] a split-and-merge meshing method is presented. Each part of the skeleton is meshed separately, then all the components are grafted together. The method is validated on a set of simple models, where the most complex junction has valence four. For complex shapes like the octopus in Figure 1, where eight limbs meet the core of the shape at the same point, it is not clear whether this method would produce a valid result. In the best case it would fail to produce the right topological structure, converting the corresponding valence nine junction into a set bifurcations. Our method produces a hex mesh that encodes the correct structure of the octopus without introducing high valence vertices, thus providing the right balance between structure and mesh complexity. Overall, we generate hexahedral meshes with higher quality (see Section 7).

In [ZBG*07] a sweeping method to mesh tubular shapes is proposed. Their method focuses on blood vessels and uses a set of templated solutions to mesh junctions, with the curve-skeleton used as a proxy to fit the best template according to the morphology of the vessel. Hexahedra are radially arranged around the skeleton, thus generating badly shaped elements near the spine. Moreover, this method works best for bifurcations (which are typical on blood vessels), while it tends to generate high valence vertices and badly shaped hexahedra when junctions with higher valence are present. Our method avoids high valence irregular vertices thus favoring better per-element quality [LSVT15].

In [LZLW15] a skeleton-based method for T-spline fitting is proposed, which can also be used for hexahedral meshing. Half-planes are employed to mesh bi-furcations and tri-furcations. This method suffers the same drawbacks as [ZBG*07]: it tends to produce overly complex meshes with high valence vertices if the skeleton contains high-valence junctions, setting a tight upper bound to the quality of the elements directly incident at them.

In [YCJL09] Yao and colleagues propose to drive the meshing process with a manually sketched curve-skeleton. Junctions are handled with a neat subdivision process. Similarly to [ULP*15], this work focuses on the generation of base domains for quadrilateral meshing and, therefore, it can be considered an alternative starting point for our method.

Grid-based methods subdivide the volume using either a regular grid or an octree and, subsequently, they move the vertices of the hexahedra intersecting the surface onto the surface itself so as to better approximate the original shape [LJLJ15, Mar09, Sch96]. Due to their simplicity and ability to mesh any object, grid-based methods are still dominant in industry. However, these methods suffer from several drawbacks: they tend to produce unnecessarily high resolu-

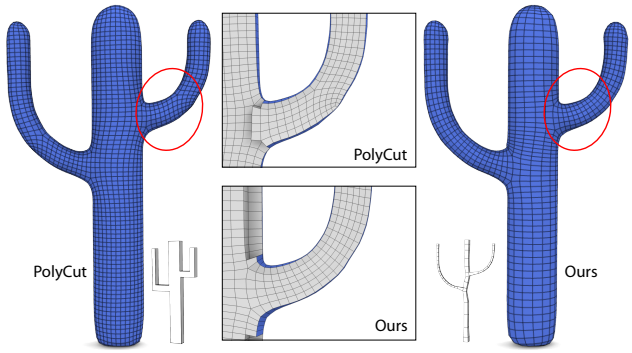


Figure 2: *Polycube mappings may force the introduction of unnecessary singularities, penalizing the alignment with the boundary of the shape (top closeup); our meshing better aligns to the limbs of the* CACTUS *(bottom closeup).*

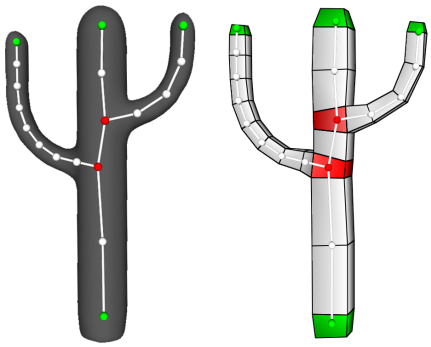


Figure 3: *We derive from the curve-skeleton of a triangle mesh (left) a volumetric decomposition in tubes (white), branching cubes (red) and terminal cubes (green). Each element in the tubular structure is a hexahedron.*

tion meshes, they are not invariant to rotation (i.e., the same volume can be meshed differently, when rotated) and they tend to push the elements of worst quality near the boundary. Our method is rotationally invariant and generates *boundary conforming* hexmeshes (Figure 2) with much less elements, also promoting high quality hexahedra near the boundary, an important requirement to ensure accurate simulations [RGRS*15].

Advancing front techniques start the mesh generation process on the surface and then move inwards [TBM96]. This approach tends to place singularities and lower quality hexahedra inside the volume. A recent example of expanding front method and a review of similar methods is provided in [KBLK14]. These methods generate high quality meshes near the boundary regions, which is a desired property for many applications. Unfortunately, they are prone to generate low quality meshes in the interior (where the fronts merge) and cannot be applied to all classes of shapes (only genus zero shapes are supported). Our method can handle complex topologies, like FERTILITY in Figure 1 and the BLOCK model in Figure 13.

Parameterization based methods map the input volume to some parametric space, where the connectivity of the mesh is generated. The inverse mapping is then used to place the corresponding vertices in the original domain. PolyCubes [THCM04] (i.e., orthogonal polyhedra) have been widely used as parametric domains because they can be trivially hex-meshed with a regular grid, generating a structured mesh [FXBH16, CLS16, HJS*14, LVS*13, GSZ11]. However, the structure of the mapping domain often causes the introduction of unnecessary singularities that penalize the alignment with the boundary (Figure 2). Other popular parameterization-based techniques associate to each point in the interior of the shape a 3D frame such that the resulting frame field is aligned to the surface of the shape. The mesh connectivity is then generated by sampling the field, with singularities occurring at its vanishing points [SRUL16, KLF14, HTWB11, NRP11]. These algorithms generate meshes that nicely adapt to the input surface, however, they do not provide any control on the structure of the mesh and tend to introduce misaligned singular vertices, resulting in a complex singular structure [LLX*12]. Our method avoids the singularity misalignment problem, resulting in coarse singularity layouts that embed the high level structure of the input shapes, a key factor to ensure high quality hexahedral meshes [GDC15]. Furthermore, there are no theoretical guarantees that a volume parameterization from a frame field admits an all-hex structure. Current methods may fail to produce a mesh depending on the type of singularities in the input field [LLX*12].

3. Pipeline overview

We propose a method to generate a full hexahedral mesh out of a tubular shape. We input a triangle mesh $\mathcal{M}$ and its curve-skeleton $\mathcal{S}$, that we use as a proxy to infer the structural properties of $\mathcal{M}$ and drive the meshing process. The result is a full hexahedral mesh $\mathcal{H}$ that embeds in its connectivity the structure of $\mathcal{S}$ and has $\mathcal{M}$ as outer boundary. We optionally apply templated schemes to control the meshing density and adapt it to the morphology of the shape, thus keeping element size variance as low as possible.

Our work builds upon the coarse quad layout generation algorithm described in [ULP*15], which we extend adding a volumetric interpretation of the tubular structure described in their paper. The method presented in [ULP*15] generates a quadrilateral mesh of tubular shapes. This mesh is obtained from a decomposition into *tubes* - cylinders with quadrilateral section that surround the skeleton curves - and *cubes* centered at both the branching and terminal nodes of the curve-skeleton (Figure 3). The quads composing each tube and each cube are derived and assembled so that the resulting mesh is *conforming* (i.e., free from T-junctions); models with loops and complex topologies are supported.

This structure lends itself to straightforward hexahedral meshing, by gridding each cube and each tube with a number of subdivisions that keep the conformity in the hex-mesh as well. Note that the resulting hexahedral mesh is *structured* by construction, because its connectivity embeds the decomposition in tubes and cubes encoded in the tubular structure, yet because each tube and each cube is meshed as a regular grid [Tau04].

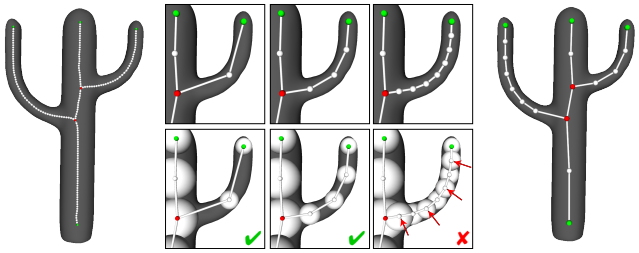


Figure 4: *We start with a dense sampling of the curve-skeleton (left) and we re-sample it by iteratively splitting half-way each of its portions. The splitting process ends when the maximal sphere centered at the new sample intersects both the spheres centered at the two end-points of the current segment (middle). The resulting coarse sampling of the skeleton (right) determines the connectivity of the final hexahedral mesh.*

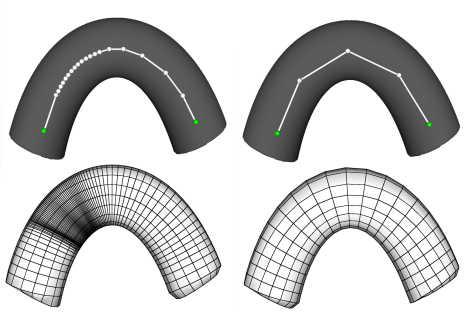


Figure 5: *The sampling of the curve-skeleton determines the density of the hexahedral meshing. Arbitrarily sampling the curve-skeleton may lead to poor meshes with badly shaped elements (left). Our sampling strategy generates good meshes and promotes isotropy (right).*

We extend the pipeline presented in [ULP*15] and adapt it to the volumetric case. Our hex-meshing strategy consists of the following steps:

1. **Skeleton resampling:** we propose a fully automatic strategy to sample the skeleton $\mathcal{S}$, in order to avoid excessively dense meshes and badly shaped elements (Section 4);
2. **Tubular structure:** we initialize mesh $\mathcal{H}$ as the tubular structure enclosing $\mathcal{S}$, placing a hexahedron at each skeleton branching node, extruding its facets along the skeleton curves and subdividing each element in order to avoid T-junctions (see [ULP*15] for details);
3. **Resolution control:** we identify changes in the local thickness of the skeleton and use a set of templates to locally adapt the meshing density to the morphology of the shape (Section 5.1);
4. **Projection:** we project the boundary of $\mathcal{H}$ onto the input shape $\mathcal{M}$, using ray casting to generate an initial poor quality map, and then we refine such map using the *abstract domains* approach described in [ULP*15, TPP*11];
5. **Finalization:** we optimize the interior of $\mathcal{H}$, carefully positioning its vertices so as to maximize per-element quality (Section 6).

In the remainder of the paper we discuss technical details regarding the sampling of $\mathcal{S}$ (1), the resolution control (3) and the hexmesh finalization (5). The initialization of the tubular structure (2) and the projection of its boundary over $\mathcal{M}$ (4) are equivalent to the ones presented in [ULP*15] and, as such, they will not be discussed here. We point the reader to the original paper for technical details regarding their implementation.

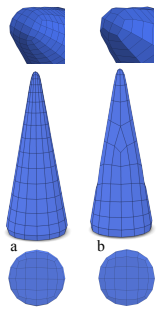
4. Skeleton resampling

Although simply gridding the tubular structure would produce a hexahedral mesh (Figure 3 right), a naive use of this technique may lead to excessively dense meshes with poorly shaped elements (see the left side of Figure 5). Indeed, the density of the mesh in the longitudinal direction is directly related with the sampling density of the underlying curve-skeleton. Expanding and subdividing the tubular structure to flood the interior of the input shape would easily result in a hex-mesh containing many inverted elements and no practical usefulness. In order to improve the mesh quality and produce well shaped elements we re-sample the curve-skeleton. For each skeleton point we assume to have the radius of its maximal sphere available. Some skeletonization algorithms already provide this information (e.g. [LS13, LGS12]); otherwise, the radius at a skeleton point can be easily estimated by measuring its distance to the input surface.

We start from a dense sampling of the curve-skeleton and, then, we sub-sample it applying arc-length parameterization to each curve of the skeleton, mapping its length in the interval $[0, 1]$. We then split the curve in the parametric space, adding a new sample point half-way, and iteratively repeat the process for the resulting sub-intervals. The stop criterion is as follows: we do not split an interval when the maximal sphere centered at a candidate sample would intersect both spheres centered at the endpoints of the current segment (Figure 4). The resulting sampling adapts to the local thickness of the shape and induces a meshing with a good isotropy (Figures 5 and 1). After re-sampling the skeleton we generate a tubular structure fully enclosing the skeleton with the method described in [ULP*15]. The result is a coarse structured hexahedral mesh $\mathcal{H}$, ready to be further subdivided and inflated to adhere to the surface of the target shape (Figure 3).

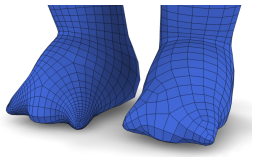
5. Resolution control

The approach described so far is capable of producing hex-meshes for any shape in our class of interest (i.e. tubular shapes); very accurate and high quality models can be obtained via the projection and finalization step described later (see Figure 1). However, depending on the morphology of the model, the elements sizes may be uneven, with the presence of high density areas that unnecessarily increase the resolution of the model. To give an example let us consider the cone-like shapes aside: the area covered by the base of the cone is much larger than the area covered by its tip, hence, if the cone is meshed with a regular grid the density of the elements on the tip will be much



higher than the elements at the base (a). In order to avoid this behavior we introduce a mechanism to adjust the resolution of the model and better adapt to the morphology of the shape, so as to keep the element size even and the resolution lower (b).

We detect the elements of the mesh where a change of resolution is needed (we call them *cones*) directly from the tubular structure derived from the curve-skeleton. As explained in Section 4, the size of each hexahedron in such structure is proportional to the local thickness of the shape, therefore the morphology of the input shape is correctly encoded in this coarse, yet easy to process, hex-mesh. We split each cone with a template subdivision that puts more sub-elements at the base and less at the tip and we propagate the resulting subdivisions throughout the whole model so as to generate a conforming hex-mesh. The effect of this approach can be seen in the inset above, where the foot of the WARRIOR is shown before (left) and after (right) the application of our cone-based resolution scaling technique. In the following subsections we illustrate how to detect cones (Section 5.1) and how to propagate the subdivision they introduce (Section 5.2).



5.1. Cone detection

Let us consider the hexahedron h depicted in Figure 6a. In order to decide whether h is a cone or not we consider the ratio between the size of facets belonging to h_{prev} and h_{next} that are opposite to the facets of h; if this ratio is > 4 we mark h as a cone and we apply the volumetric subdivision schemes depicted in the right part of Figure 7, each of which is capable of scaling the resolution of the mesh by a factor of 4. Note that these subdivision patterns are just a tiny subset of all the possible ways to scale the resolution; as pointed out in [TPSH14] the problem of enumerating an exhaustive list of subdivision schemes is wide open.

In our experiments we observed that using too many cones may result in a very high resolution mesh; we therefore restrict our cone detection strategy only to the terminal branches of the curve-skeleton and we limit the presence of cones at up to two for each skeleton branch. If more than two candidate cones are found along a skeleton branch we rank them according to the ratio described above and we select the two top ranked elements. Specifically, we enable the presence of one 1×1 to 2×2 and one 2×2 to 4×4 cones, thus achieving a maximum scaling factor of 4^2 (see Figure 9). Our choice to restrict to at most two cones for each terminal branch is

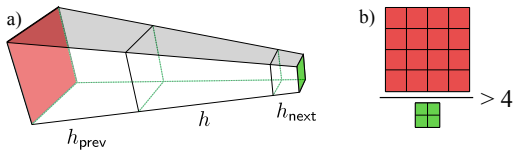


Figure 6: *A hexahedron h (a) is a "cone" if the ratio between the areas of the opposite facets of its two adjacent hexahedra (here in red and green) is larger than 4 (b).*

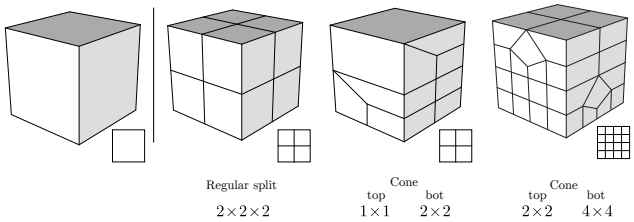


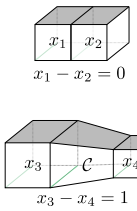
Figure 7: *The volumetric splitting schemes we use in our meshing algorithm. From left to right: a single hexahedron; a regularly split hexahedron; a cone that scales from* 1×1 *to* 2×2*; a cone that scales from* 2×2 *to* 4×4*.*

justified from the fact that the class of objects we are interested in (i.e., biological structures like humanoids, plants and animals) tend to have a thicker core and thinner terminal limbs, thus requiring a resolution scale only at the peripheries of the shape.

5.2. Subdivisions propagation

We propagate the subdivisions induced by the cones throughout the mesh by solving an Integer Linear Programming (ILP) problem. We associate to each regular element of the hex-mesh (i.e., not a cone) a variable x that represents the number of splits necessary to achieve mesh conformity. Regular elements are always split with the $2\times2\times2$ pattern depicted in Figure 7, iteratively applied as many times as indicated by the associated integer variable.

Specifically, for each pair of adjacent elements we require the number of splits to be equal (inset aside, top), whereas for each pair of hexahedra adjacent to the same cone $\mathcal{C}$ we require the element at the base to be split once more than the element at the tip (inset aside, bottom). Furthermore, since we are solving for the number of splits of each element, we ask each variable x to be positive. This results in the following integer linear programming problem



$$\begin{aligned} \min\ & AX = b \\ & X \geq 0 \\ & X \in \mathbb{I}^n \end{aligned}$$

with n being the number of regular elements in the mesh (here the cones do not count), A being a sparse $n\times n$ coefficient matrix and b being a sparse vector. We solve such problem with Gurobi [Gur]. Note that since we admit cones only in the terminal limbs of a shape no cone will appear in a loop; consequently such system will always admit a solution regardless the topology of the shape, as discussed in [ULP*15]. In Figure 8 we show a 2D example of our propagation system; real examples can be seen in Figure 9, where both DINO and DINOPET are depicted.

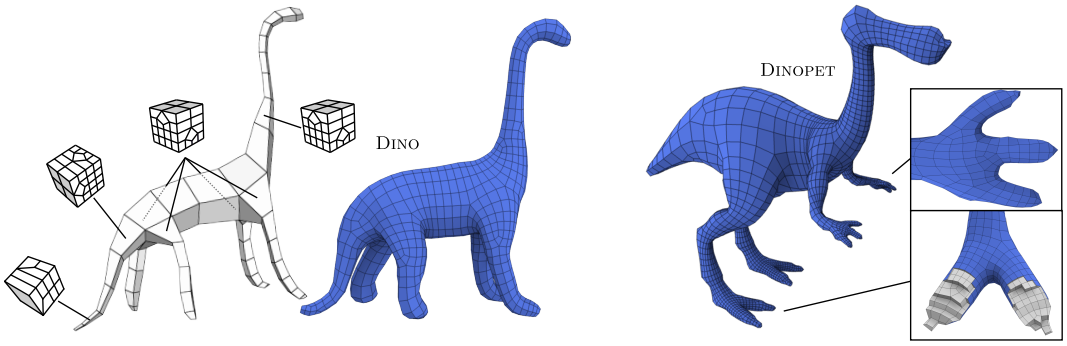


Figure 9: *Left: a hexahedral mesh of the* DINO *model obtained using our resolution scaling scheme; the volumetric tubular structure used to derive the meshing process. If a limb contains only one cone (legs, neck) we apply a* 2×2 *to* 4×4 *subdivision scheme; if there are two cones along the same limb (tail) we place one* 1×1 *to* 2×2 *and one* 2×2 *to* 4×4 *subdivision schemes, thus achieving a maximum scaling factor of* 4^2*. Right: another example of our resolution scaling technique applied to the* DINOPET.

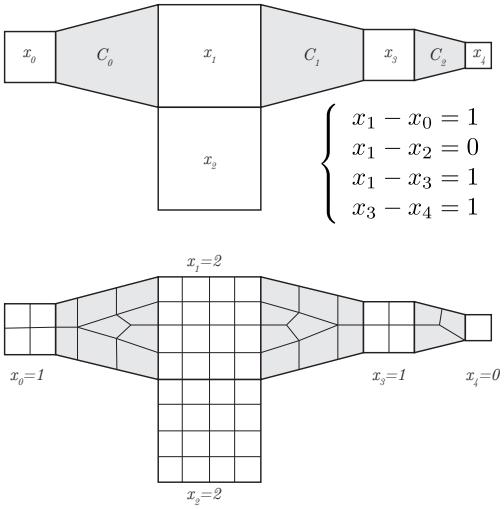


Figure 8: *A 2D example of our subdivision propagation system. We ask adjacent elements (e.g.,* x_1, x_2*) to split the same number of times, and elements at the base of a cone (e.g.,* x_1, x_3*) to split once more than the elements at the tip of the same cone (*x_0, x_3, x_4*). By solving the resulting ILP problem we know how many times we need to split each element in the mesh to make it conforming.*

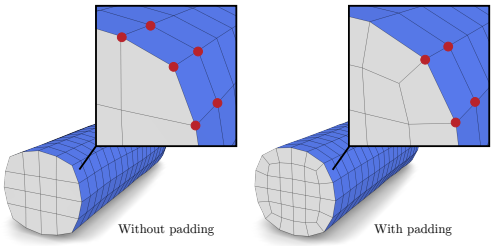


Figure 10: *Prior to padding boundary elements may have more than one facet exposed on the surface. If these facets are coplanar the element will become degenerate, with no chances of optimization as at least six vertices will be constrained on the surface. We therefore add a padding layer. This ensures that each element will have at most one facet exposed (four vertices), thus leaving enough degrees of freedom for the subsequent mesh optimization.*

6. Projection and Finalization

As already mentioned in Section 3 the projection phase is the same used in [ULP*15]. We briefly recap it here just for completeness. We first use ray-casting to inflate the boundary of the hexa mesh, so as to map it to the input triangle-mesh; we then optimize the resulting mapping using the *abstract domains* technique introduced in [TPP*11]. Please refer to [ULP*15, TPP*11] for further details.

Once the surface of $\mathcal{H}$ has been mapped onto the target surface $\mathcal{M}$, we optimize the position of internal vertices, by minimizing the following quadratic energy

$$\sum_{i}^{n} \sum_{j \in N(i)} \|v_i - v_j\|^2$$

Here $N(i)$ is the *volumetric* one ring of the vertex v_i. Minimizing the energy above requires the resolution of a sparse linear system $Ax = b$ in the least square sense, according to the normal equation $A^T Ax = A^T b$.

The hexahedral meshes so generated have good average quality but can, and in general do, contain *inverted* elements, that is, non-convex hexahedra [Knu00]. Even a single inverted element makes a mesh unusable for applications [PTS*07], therefore a further optimization step aimed at removing inverted elements is needed. Since the focus of this work is the generation of a high quality topology, we rely on standard optimization tools for the improvement of per element quality. Notice that separating the generation of the connectivity from the optimization of the embedding is a classical approach in hex-mesh generation [LSVT15].

In order to improve mesh quality we first add a *pillowing* (or

padding) layer, extruding the surface quads to form a shell of hexahedra surrounding the outer surface of the hexahedral mesh [GSZ11]. After pillowing, each element of the mesh will have at most four vertices on the surface. Consequently, at least four vertices per element will be free to move in the interior, guaranteeing enough degrees of freedom for the subsequent quality optimization (Figure 10). We then apply the edge-cone rectification algorithm [LSVT15] to remove all the inverted elements from the mesh and improve on both minimum and average quality (see Table 1). For the edge-cone rectification algorithm we always used the same parameters, that is: automatic estimation of the target edge-length, attraction to the surface weight (α) equal to 20 and attraction to the sharp features (β) equal to 0.

7. Results

We have implemented our algorithm as a single threaded C++ application and we run our tests on a MacBook Air equipped with a 1.7GHz Intel Core i5 and 4GB of RAM. The running times of our method vary from 1.5 seconds for a simple shape like CACTUS to approximately 1 minute for a complex shape like WARRIOR. Alternative meshing strategies such as [HJS*14, LVS*13] are one order of magnitude slower.

For the computation of the curve-skeletons we used different algorithms, specifically: ARMADILLO, BIG BUDDY, BLOCK, CACTUS, CLEF, DINOPET, DINOSAUR, OCTOPUS, SANTA, BLOOD, VESSEL and WARRIOR were skeletonized using the 3D-from-2D approach described in [LS15, LS13, LGS12]; FERTILITY and ROCKER ARM were skeletonized using the mean curvature approach described in [TAOZ12]. In the latter case, we manually simplified the skeleton, merging nearby branching nodes to better reflect the logical structure of the shape (we used Skeleton Lab [BMU*16] for each such editing). This operation was not necessary for the skeletons computed with [LS15, LS13, LGS12] as they already embed heuristics for the automatic collapse of spurious branches.

We tested our method on a wide range of objects (Figures 1, 2, 9, 12); next to each model we show the tubular structure we used to drive the meshing process. The meshes produced by our method embed a volume decomposition that reflects the logical structure of the input shape and naturally align with its main features. As acknowledged in [Bla01] a good alignment with the features of a shape leads to superior results in the simulations. Furthermore, recent studies have shown that simplifying the singularity graph of a hexahedral mesh by aligning its singular vertices helps to keep the resolution lower and at the same time improves the mesh quality [GDC15]. Our method builds upon the quad layout generation algorithm proposed in [ULP*15] so it naturally aligns singular vertices in a meaningful way, providing as-coarse-as-possible singularity layouts that turn into high quality, boundary conforming, hexahedral meshes (see yellow lines in the top closeup in Figure 1).

We summarize our results in Table 1, where we also compare against the skeleton-based method proposed in [LLD12], two PolyCube-based methods [HJS*14, LVS*13], one frame field-based method [LLX*12], an Octree-based method [Mar09] and the recently published Generalized Sweeping [GMD*15] and CVIF [LJLJ15] methods. For each model we report: the mesh resolution

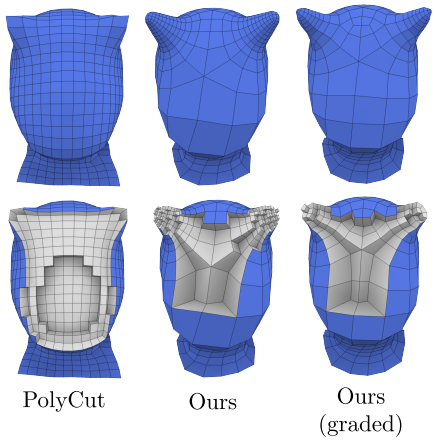


Figure 11: *The head of the* WARRIOR *meshed with PolyCut (left), our method without applying the resolution scaling step (center) and applying it (right).*

(for both input and output), and the minimum and average quality of the output hex-mesh and the average distance from the input surface. We evaluate quality using the Scaled Jacobian (SJ), a popular metric that measures the deviation of each element from a perfect cube [PTS*07]; the SJ is defined within the range $[-1, 1]$ with one being optimal and negative values denoting inverted elements. None of the models shown throughout the paper contains inverted elements (i.e., min SJ > 0), this is a fundamental minimum requirement for many applications involving hexahedral meshes [Bla01].

Since we use [LSVT15] in the final step of our method (Section 6), for the sake of a fair comparison we optimized the meshes of our competitors with the same technique, whenever possible. Exceptions to this rule are: the Generalized Sweeping [GMD*15] and Frame Field [LLX*12] approaches, for which we report the quality from the orginal papers as with [LSVT15] it was impossible to improve any further; and CVIF [LJLJ15] and the skeleton-based approaches [LLD12], for which we did not have the geometry available. Notice that both [LJLJ15] and [LLD12] already employ some optimization strategy to finalize their meshes, therefore we believe that the values reported in Table 1 truly reflect the potential of the connectivity generated by their meshing strategies.

From the qualitative point of view the performance of our algorithm matches the parameterization-based methods [HJS*14, LVS*13, LLX*12] and outperforms Generalized Sweeping [GMD*15], grid-based [LJLJ15, Mar09] and skeleton-based [LLD12] algorithms. Our method achieved average SJ above 0.9 for the majority of the models we produced, outperforming previous skeleton-based methods like [LLD12], whose hexahedral meshes hardly exceed 0.8 average SJ.

Another important feature that emerged from our tests is the ability to keep the resolution low, an important criterion for fast simulations. As can be noticed in Table 1 we produced the coarsest hex-meshes in the majority of the comparisons. From this point of view the worst performances come from the Octree [Mar09] and voxel-based CVIF [LJLJ15] methods.

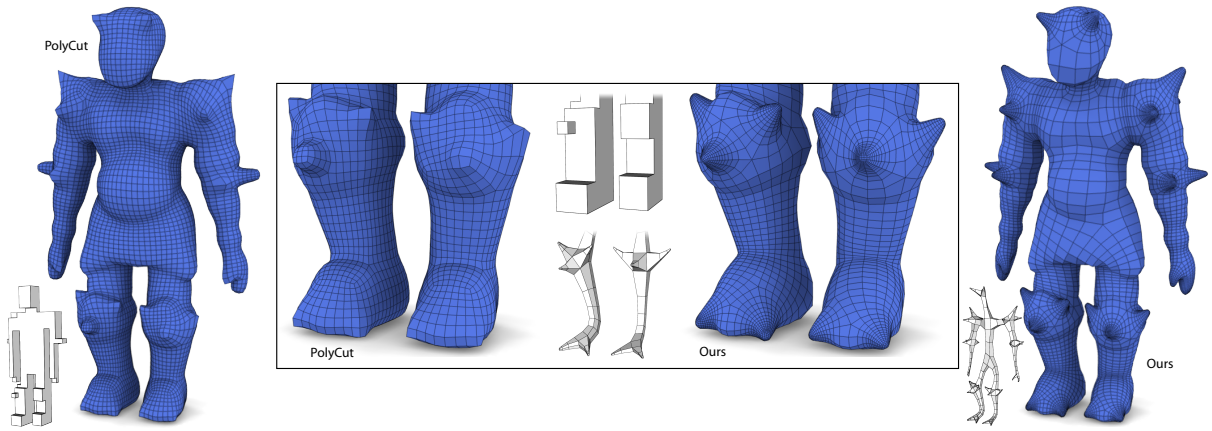


Figure 12: *Mapping to an axis aligned domain strongly limits the ability to represent off axis features; most of the spikes of the* WARRIOR *are not caught by the polycube, resulting in a poor meshing that hardly matches the original geometry (left). Our tubular structure nicely follows each and every spike, generating a better meshing (right).*

We also propose some visual comparison against a polycube-based method (i.e., PolyCut [LVS*13]). As one can notice, in Figure 2, polycube-based methods may not align with the surface of the model, placing unnecessary singularities that prevent the edges to align with the limbs of the CACTUS. This property, called *surface conformity*, serves to promote the placement of high quality elements close to the boundary of the model and it is an important factor to ensure accurate simulations [RG11]. The connectivity generated by our method nicely aligns with both the limbs and the core of the shape enabling the placement of high quality elements nearby the boundary of the shape. Our method is also able to generate a connectivity that nicely fits the assembly of spikes in the knees, elbows and shoulders of the WARRIOR (Figure 12). We note that, because of the rigid structure of the parametric domain, such a meshing is impossible to achieve with a polycube-based method. As it can be noticed in the closeup in Figure 12, some of the spikes are not caught by the polycube, resulting in a hexahedral mesh that hardly fits the target geometry.

In Figure 11 we make a sample visual comparison also with our method once the resolution scaling is applied. As one can see from the picture on the right, our complete method obtains a reasonable compromise between element regularity (for which PolyCut is optimal) and alignment with the features (for gaining the optimum on this we should not apply the reduction scheme).

Finally, although out of the scope of our method, we run some preliminary tests on mechanical parts. As one can notice in Figure 13, we have been able to produce full hexahedral meshes for our test models, but we still fail at aligning the edge flow with the sharp edges and features of the shapes. In the future, we plan to improve our meshing strategy by taking into account the alignment to sharp features, so as to be able to embrace a broader range of shapes.

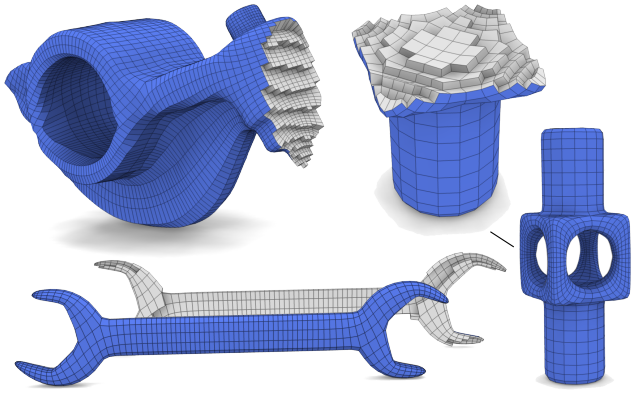


Figure 13: *Some preliminary test on three mechanical parts:* ROCKERARM *(top left),* BLOCK *(right) and* CLEF *(bottom). Although out of the scope of this work our method could produce full hexahedral meshes for each model, but it still fails at aligning the meshing to sharp features; something that we plan to work on in the future.*

8. Conclusions

We have introduced a skeleton-based algorithm for the automatic generation of structured hexahedral meshes of tubular shapes, and we have also presented novel techniques for the control of the resolution of the hexahedral mesh, both across and along the skeleton curves. To reach this goal we exploit the properties of the curve-skeleton, using it as a proxy to derive structural information about 3D shapes, as in [ULP*15]. We then use such information to construct volumetric meshes that nicely align with the branching structure of the target shape. The method is easy to code. It does not require any user parameter and it generates quality meshes for any 3D model that admits a skeletal representation.

8.1. Limitations and further works

This method is inherently limited in its scope by the class of shapes that admit a skeletal representation. Although this is not a real lim-

Model	#Tris	#Hexa	avg/min SJ	Avg dist
ARMADILLO				
PolyCut†		30K	**.90**/.14	
Ours	331K	**4K**	.88/**.21**	3.7×10^{-5}
BIG BUDDY				
Ours	27K	15K	.90/.32	4.5×10^{-5}
BLOCK				
PolyCut†		**3K**	.87/.25	
Octree-based†		20K	**.91**/.27	
Ours	5K	4K	.81/**.36**	8.7×10^{-6}
CACTUS				
PolyCut†		8K	**.94**/.42	
Ours	11K	**4K**	.92/**.52**	2.1×10^{-6}
CLEF				
Octree-based†		10K	**.90/.34**	
Ours	3K	**2K**	.86/.29	2.9×10^{-5}
DINOPET				
Ours	9K	18K	.92/.18	5.3×10^{-5}
DINOSAUR				
Ours	47K	9K	.91/.41	4.7×10^{-5}
FERTILITY				
ℓ_1 PolyCubes†		18K	**.94**/.42	
PolyCut†		54K	.86/.34	
Skel-based§		16K	.75/.08	
CVIF§		107K	.90/.04	
Frame-field‡		14K	.91/.35	
Gen.Sweep‡		20K	.82/.18	
Ours	33K	**8K**	.90/**.50**	2.6×10^{-5}
OCTOPUS				
Ours	66K	5K	.88/.11	4.4×10^{-4}
ROCKERARM				
ℓ_1 PolyCubes†		24K	**.96/.59**	
PolyCut†		57K	**.96**/.58	
Frame-field †		**11K**	.94/.57	
Gen.Sweep‡		**11K**	.83/.11	
CVIF§		63K	.90/.06	
Ours	20K	19K	.91/.16	6.2×10^{-5}
SANTA				
Skel-based§		**15K**	.72/.08	
CVIF§		73K	.88/.04	
Ours	13K	26K	**.94/.37**	2.2×10^{-4}
BLOOD VESSEL				
Ours	60K	5K	.88/.32	9.0×10^{-4}
WARRIOR				
PolyCut†		24K	**.94**/.14	
Ours	27K	**19K**	.90/**.29**	8.7×10^{-5}

† optimized with [LSVT15]
‡ data from the original paper, we could not improve on quality any further using [LSVT15]
§ data from the original paper, models not available

Table 1: *Summary of our results. From left to right: number of input triangles, number of output hexahedra, average and minimjm Scaled Jacobian (compute using the Verdict Library [SEK*07]), average deviation from the input surface (measure using Metro [CRS98]). We compare against: ℓ_1 PolyCubes [HJS*14], PolyCut [LVS*13], Frame-field based [LLX*12], Skeleton-based [LLD12], Generalized Sweeping [GMD*15], CVIF [LJLJ15] and Octree-based [Mar09] all-hexahedral meshing techniques. For each model, we highlight in bold both the lowest resolution and the highest min/avg quality.*

itation for biological shapes like humanoids, animals, vessels and plants, we would like to be more general and embrace a wider class of shapes. We are therefore looking for other shape descriptors or shape understanding processes that can be exploited to derive structural information about general 3D shapes, to be used to accomplish tasks like surface and volume remeshing.

Similarly to polycubes, the hexahedral meshes generated by our method have a simple block structure that does not conform to a frame field [FXBH16, LLX*12]. As a consequence, the twisting component along the skeleton curves can hardly be controlled, possibly leading to poor meshing results.

The method also inherits the limitations of [ULP*15] regarding the alignment to sharp features. As it is mainly intended for biologically-inspired shapes, at the moment the preservation of sharp features is not addressed.

References

[ACSYD05] ALLIEZ P., COHEN-STEINER D., YVINEC M., DESBRUN M.: Variational tetrahedral meshing. *ACM Transactions on Graphics 24*, 3 (2005), 617–625. 2

[Bla01] BLACKER T.: Automated Conformal Hexahedral Meshing Constraints, Challenges and Opportunities. *Engineering with Computers 17*, 3 (2001), 201–210. 2, 7

[BMU*16] BARBIERI S., MELONI P., USAI F., SPANO L. D., SCATENI R.: An Interactive Editor for Curve-Skeletons: SkeletonLab. *Computer & Graphics to appear* (2016). 7

[CLS16] CHERCHI G., LIVESU M., SCATENI R.: Polycube Simplification for Coarse Layouts of Surfaces and Volumes. *Computer Graphics Forum 35*, 5 (2016), 11–20. 3

[CRS98] CIGNONI P., ROCCHINI C., SCOPIGNO R.: Metro: measuring error on simplified surfaces. *Computer Graphics Forum 17*, 2 (1998), 167–174. 9

[CSM07] CORNEA N. D., SILVER D., MIN P.: Curve-Skeleton Properties, Applications, and Algorithms. *IEEE Transactions on Visualization and Computer Graphics 13*, 3 (2007), 530–548. 2

[FXBH16] FANG X., XU W., BAO H., HUANG J.: All-hex meshing using closed-form induced polycube. *ACM Transactions on Graphics 35*, 4 (2016), 124. 3, 9

[GDC15] GAO X., DENG Z., CHEN G.: Hexahedral Mesh Re-parameterization from Aligned Base-complex. *ACM Transactions on Graphics 34*, 4 (2015), 142:1–142:10. 2, 3, 7

[GMD*15] GAO X., MARTIN T., DENG S., COHEN E., DENG Z., CHEN G.: Structured volume decomposition via generalized sweeping. *Visualization and Computer Graphics, IEEE Transactions on PP*, 99 (2015), 1–1. 7, 9

[GSZ11] GREGSON J., SHEFFER A., ZHANG E.: All-Hex Mesh Generation via Volumetric PolyCube Deformation. *Computer Graphics Forum 30*, 5 (2011), 1407–1416. 3, 7

[Gur] GUROBI: Optimizer 6.5. http://www.gurobi.com/. 5

[HJS*14] HUANG J., JIANG T., SHI Z., TONG Y., BAO H., DESBRUN M.: ℓ1-Based Construction of Polycube Maps from Complex Shapes. *ACM Transactions on Graphics 33*, 3 (2014), 25:1–25:11. 3, 7, 9

[HTWB11] HUANG J., TONG Y., WEI H., BAO H.: Boundary Aligned Smooth 3D Cross-frame Field. *ACM Transactions on Graphics 30*, 6 (2011), 143:1–143:8. 3

[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*, Sarrate J., Staten M., (Eds.). Springer International Publishing, 2014, pp. 147–164. 2, 3

[KLF14] KOWALSKI N., LEDOUX F., FREY P.: Block-structured hexahedral meshes for cad models using 3d frame fields. *Procedia Engineering 82* (2014), 59–71. 3

[Knu00] KNUPP P. M.: Achieving finite element mesh quality via optimization of the Jacobian matrix norm and associated quantities. Part I-a framework for surface mesh optimization. *International Journal for Numerical Methods in Engineering 48*, 3 (2000), 401–420. 6

[LGS12] LIVESU M., GUGGERI F., SCATENI R.: Reconstructing the Curve-Skeletons of 3D Shapes Using the Visual Hull. *IEEE Transactions on Visualization and Computer Graphics 18*, 11 (2012), 1891–1901. 4, 7

[LJLJ15] LIN H., JIN S., LIAO H., JIAN Q.: Quality guaranteed all-hex mesh generation by a constrained volume iterative fitting algorithm. *Computer-Aided Design 67* (2015), 107–117. 2, 7, 9

[LLD12] LIN H., LIAO H., DENG C.: *Filling Triangular Mesh Model with All-Hex Mesh by Volume Subdivision Fitting*. Tech. Rep. TR-ZJUCAD-2012-002, State Key Lab of CAD, Zhejiang University, 2012. 2, 7, 9

[LLX*12] LI Y., LIU Y., XU W., WANG W., GUO B.: All-hex Meshing Using Singularity-restricted Field. *ACM Transactions on Graphics 31*, 6 (2012), 177:1–177:11. 3, 7, 9

[LS13] LIVESU M., SCATENI R.: Extracting Curve-skeletons from Digital Shapes Using Occluding Contours. *The Visual Computer 29*, 9 (2013), 907–916. 4, 7

[LS15] LIVESU M., SCATENI R.: Practical Medial Axis Filtering for Occlusion-Aware Contours. In *Smart Tools and Apps for Graphics - Eurographics Italian Chapter Conference* (2015), The Eurographics Association. 7

[LSVT15] LIVESU M., SHEFFER A., VINING N., TARINI M.: Practical Hex-Mesh Optimization via Edge-Cone Rectification. *ACM Transactions on Graphics 34*, 4 (2015), 141:1–141:11. 2, 6, 7, 9

[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 (2013), 171:1–171:12. 3, 7, 8, 9

[LZLW15] LIU L., ZHANG Y., LIU Y., WANG W.: Feature-preserving T-mesh construction using skeleton-based polycubes. *Computer-Aided Design 58* (2015), 162–172. 2

[Mar09] MARECHAL L.: Advances in Octree-Based All-Hexahedral Mesh Generation: Handling Sharp Features. In *Proceedings of the 18th International Meshing Roundtable*, Clark B., (Ed.). Springer Berlin Heidelberg, 2009, pp. 65–84. 2, 7, 9

[NMK*06] NEALEN A., MÜLLER M., KEISER R., BOXERMAN E., CARLSON M.: Physically based deformable models in computer graphics. *Computer Graphics Forum 25*, 4 (2006), 809–836. 2

[NRP11] NIESER M., REITEBUCH U., POLTHIER K.: CubeCover- Parameterization of 3D Volumes. *Computer Graphics Forum 30*, 5 (2011), 1397–1406. 3

[PTS*07] PÉBAY P. P., THOMPSON D., SHEPHERD J., KNUPP P., LISLE C., MAGNOTTA V. A., GROSLAND N. M.: New Applications of the Verdict Library for Standardized Mesh Verification Pre, Post, and End-to-End Processing. In *Proceedings of the 16th International Meshing Roundtable*. 2007, pp. 535–552. 6, 7

[RG11] RUIZ-GIRONÉS E.: *Automatic Hexahedral Meshing Algorithms: from Structured Meshes to Unstructured Meshes*. PhD thesis, Polytechnic University of Catalonia, 2011. 2, 8

[RGRS*15] RUIZ-GIRONÉS E., ROCA X., SARRATE J., MONTENEGRO R., ESCOBAR J.: Simultaneous untangling and smoothing of quadrilateral and hexahedral meshes using an object-oriented framework . *Advances in Engineering Software 80* (2015), 12–24. 3

[Sch96] SCHNEIDERS R.: A grid-based algorithm for the generation of hexahedral element meshes. *Engineering with Computers 12*, 3-4 (1996), 168–177. 2

[SEK*07] STIMPSON C., ERNST C., KNUPP P., PÉBAY P., THOMPSON D.: The verdict library reference manual. *Sandia National Laboratories Technical Report* (2007). 9

[SRUL16] SOKOLOV D., RAY N., UNTEREINER L., LÉVY B.: Hexahedral-dominant meshing. *ACM Transactions on Graphics 35*, 5 (2016), 157:1–157:23. 3

[TAOZ12] TAGLIASACCHI A., ALHASHIM I., OLSON M., ZHANG H.: Mean Curvature Skeletons. *Computer Graphics Forum 31*, 5 (2012), 1735–1744. 7

[Tau01] TAUTGES T. J.: The generation of hexahedral meshes for assembly geometry: survey and progress. *International Journal for Numerical Methods in Engineering 50*, 12 (2001), 2617–2642. 2

[Tau04] TAUTGES T. J.: MOAB-SD: Integrated structured and unstructured mesh representation. *Engineering with Computers 20*, 3 (2004), 286–293. 2, 3

[TBM96] TAUTGES T. J., BLACKER T., MITCHELL S. A.: The Whisker Weaving Algorithm: a connectivity-based method for constructing all-hexahedral finite element meshes. *International Journal for Numerical Methods in Engineering 39*, 19 (1996), 3327–3349. 3

[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. 2

[THCM04] TARINI M., HORMANN K., CIGNONI P., MONTANI C.: PolyCube-Maps. *ACM Transactions on Graphics 23*, 3 (2004), 853–860. 3

[TPP*11] TARINI M., PUPPO E., PANOZZO D., PIETRONI N., CIGNONI P.: Simple quad domains for field aligned mesh parametrization. *ACM Transactions on Graphics 30*, 6 (2011), 142:1–142:12. 4, 6

[TPSH14] TAKAYAMA K., PANOZZO D., SORKINE-HORNUNG O.: Pattern-Based Quadrangulation for N-Sided Patches. *Computer Graphics Forum 33*, 5 (2014), 177–184. 5

[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 Trans. Graph. 35*, 1 (Dec. 2015), 6:1–6:13. 2, 3, 4, 5, 6, 7, 8, 9

[YCJL09] YAO C.-Y., CHU H.-K., JU T., LEE T.-Y.: Compatible quadrangulation by sketching. *Computer Animation and Virtual Worlds 20*, 2-3 (2009), 101–109. 2

[ZBG*07] ZHANG Y., BAZILEVS Y., GOSWAMI S., BAJAJ C. L., HUGHES T. J.: Patient-specific vascular {NURBS} modeling for isogeometric analysis of blood flow. *Computer Methods in Applied Mechanics and Engineering 196*, 29-30 (2007), 2943–2959. 2