Explicit Cylindrical Maps for General Tubular Shapes

M. Livesu, M. Attene, G. Patanè, M. Spagnuolo
CNR-IMATI, Italy

Abstract

A solid cylindrical parameterization is a volumetric map between a tubular shape and a right cylinder embedded in the polar coordinate reference system. This paper introduces a novel approach to derive smooth (i.e., harmonic) cylindrical parameterizations for solids with arbitrary topology. Differently from previous approaches our mappings are both fully explicit and bi-directional, meaning that the three polar coordinates are encoded for both internal and boundary points, and that for any point within the solid we can efficiently move from the object space to the parameter space and vice-versa. To represent the discrete mapping, we calculate a tetrahedral mesh that conforms with the solid's boundary and accounts for the periodic and singular structure of the parametric domain. To deal with arbitrary topologies, we introduce a novel approach to exhaustively partition the solid into a set of tubular parts based on a curve-skeleton. Such a skeleton can be either computed by an algorithm or provided by the user. Being fully explicit, our mappings can be readily exploited by off-the-shelf algorithms (e.g., for iso-contouring). Furthermore, when the input shape is made of tubular parts, our method produces low-distortion parameterizations whose iso-surfaces fairly follow the geometry in a natural way. We show how to exploit this characteristic to produce high-quality hexahedral and shell meshes.

Keywords: Solid modeling; generalized cylinder; cylindrical map; volumetric parameterization; hexahedral meshing

1. Introduction

Technology is pushing fast the borders of computer graphics and contributes to increase the need for a tighter integration of design and simulation, both in engineering domains and in very unspecialized areas. Shape models available today are extremely sophisticated in terms of the realism they can visually convey, but are still far from being suitable to enter a continuous pipeline of processing that goes through design, simulation, and physical realization [1]. Volume-based parameterization methods are intriguing for their potential to support the growth of novel unified representation modalities: internal properties of solid objects can be modelled and manipulated explicitly, and simulation of physical properties can rely on a natural discretization of the geometric domain.

Various volumetric maps have been proposed in recent years to generate solid representations of 3D shapes, ranging from piece-wise linear meshes made of tetrahedra [2], hexahedra [3] or mixed-elements [4], to continuous CAD-like representations such as trivariate splines [5, 6, 7].

An extremely popular approach to volumetric maps uses harmonic functions as generatrices [8, 9, 10]. This choice is motivated by their regularity, well behavior, and the possibility to explicitly control singular points. On the downsides, a common issue of these methods is the need to have the user in the loop, as they require a manual placing of the local maxima and minima. Controlling a harmonic field by manually placing the right amount of singularities just in the right position is known in literature to be an extremely complex task, even for experienced users [11, 12].

In this paper, we focus on solid cylindrical parameterizations (Figure 1). Our maps are driven by three harmonic functions, which represent the height, radius, and azimuth polar coordinates of a parametric cylinder. Polar cylindrical maps are quite a natural choice for tubular shapes, as they encode the absolute position of each point inside the object in terms of its relative location with respect to the inner axis. Indeed, polar cylindrical maps provide a direct control of the twist component and enable an intuitive modeling of composite layered objects such as bones [13]. However, they are restricted to simple shapes composed of a single tube, do not scale on complex shapes with higher genus or multiple branches, and are intrinsically difficult to encode in the vertices of a mesh due to the singular and periodic structure of the polar parametric domain (Section 3).

The state-of-the-art in the field [13] proposes an implicit representation of the map for the interior of the object, storing the azimuth coordinates only for the boundary vertices. The interior of the map is evaluated on-the-fly, with a cumbersome procedure that first extracts a cross-section orthogonal to the axis, and then traces an integral curve from the boundary to the selected point. No method is described to navigate the map in the opposite direction (i.e., locating object points in the parametric domain). Though effective, this approach is extremely inefficient from a computational

standpoint (30 minutes for the femur shown in Figure 12 — our pipeline runs in seconds). Furthermore, being implicit the map cannot be exported to a file for later use, shared with other users and, most importantly, cannot be used by standard algorithms, which would expect it to be in explicit form, that is, encoded in the vertices of a volumetric mesh (e.g., for iso contouring [14] or optimization [15]).

We extend the volumetric cylindrical maps proposed in [13] and also address common issues in the generation of harmonic maps, such as the manual placement of singular vertices. Specifically, we propose an automatic method to generate solid maps of complex tubular shapes with arbitrary topology. Our contributions are twofold:

- we propose a neat method to discretize the interior of a tubular shape with a tetrahedral mesh, which accounts for the periodic and singular structure of the polar parametric domain. The map can be easily attached to the vertices, and linearly extended in the interior of each tetrahedron. In this form, the map can be readily navigated in both directions and exploited by off-the-shelf algorithms;

- we remove the user from the loop, using a shape proxy (i.e., the curve-skeleton) to drive the decomposition of a complex tubular shape of arbitrary genus into a set of disjoint atomic tubes, each corresponding to exactly one of the bones. The algorithm automatically places the singularities of the field, and is oblivious to the particular skeleton being used. The only requirement is that the skeleton is homotopic with respect to the object and it is completely contained in its volume. To the best of our knowledge, our multi-cylinder approach is the first method capable of extending cylindrical maps to shapes with complex topologies. In a sense, it can be seen as the equivalent of the multi-block parameterization often used for cuboidal maps [16, 11].

With our approach, any volume enclosed by a shape which is well described by its curve-skeleton can be parametrized, with various domain discretization choices available thereafter. The input curve skeleton can be either computed by an algorithm [17, 18, 19, 20] or sketched by a user [21]. In a sense, the curve-skeleton becomes a tool to control the map (Figure 10) — note that for some professionals, such as animators, defining a skeleton is much more natural than specifying critical points. We demonstrate the usefulness and intuitiveness of our tool in classical solid modeling problems, such as hexahedral meshing and shell meshing.

2. Related work

Our work relates with the following research areas.

Surface Cylindrical Maps. Cylindrical parameterizations for surfaces have been extensively studied in literature. In [22, 23, 24], three methods are presented to map a surface with two boundaries to an open cylinder. Huysmans *et al.* [25] compared cylindrical and spherical surface maps for statistical medical shape modeling, emphasizing the importance of using cylindrical parametric domains for elongated tubular bones. Kalberer *et al.* [26] proposed a stripe parameterization method for tree-like genus zero shapes based on branched coverings. These methods are restricted to surfaces only and are mostly inspired by problems arising in medicine, such as the analysis of blood vessel and bone shapes. We propose a volumetric approach and consider *general* tubular shapes; i.e., any shape that can be decomposed into atomic tubes having two bases and a lateral surface, with no restrictions on topology, morphology, and on the way tubes are attached to one another.

Volumetric Maps. Mappings between a shape and a solid parametric domain are at the core of many recent modeling techniques. The generatrices of the map can be various. Popular choices are harmonic [10] and bi-harmonic functions [8], but the map can also be generated through a continuous deformation that goes from the shape to a parametric space composed of multiple face-adjacent blocks [3, 27]. The latter case is possibly the most flexible, but a proper parametric domain must be computed *a priori*. Scientific literature offers a plethora of different methods to generate well-shaped multi-block parametric domains [28, 6, 11, 29, 7, 9]. Though very general, for tubular shapes cylindrical domains are still a better solution as they introduce less singularities, better align to the local curvatures and naturally encode each point inside the object with respect to the inner axis. We outperform the state of the art in the field [13] by providing an explicit representation of the cylindrical map and by supporting general tubular shapes with arbitrary topology (Section 6).

Tubular Shapes. Decomposing a shape into a set of simpler primitives is a fundamental problem in geometry processing. In particular, tubular (or cylindrical) decompositions are widely used for shape approximation and modeling. Methods such as [30, 31] aim to recognize the cylindrical components of a shape, though are not applicable to our specific problem since they may leave some portions of the shape with weak cilindricity outside of the decomposition. Similarly, Wu and Lui [32] propose a skeleton-driven tube decomposition, but their method leaves some unassigned voids around the branching nodes of the skeleton. Antiga *et al.* [33] proposed an effective method specific for bifurcating blood vessels. Their method does not scale on general shapes. The work of Thiery *et al.* [34] is very similar in spirit to ours, as they decompose a tubular object into a set of cylinders and then use a surface cylindrical map to generate an analytic curve-skeleton. Our method can be seen as its volumetric extension. A number of methods in literature propose skeleton-driven modeling techniques

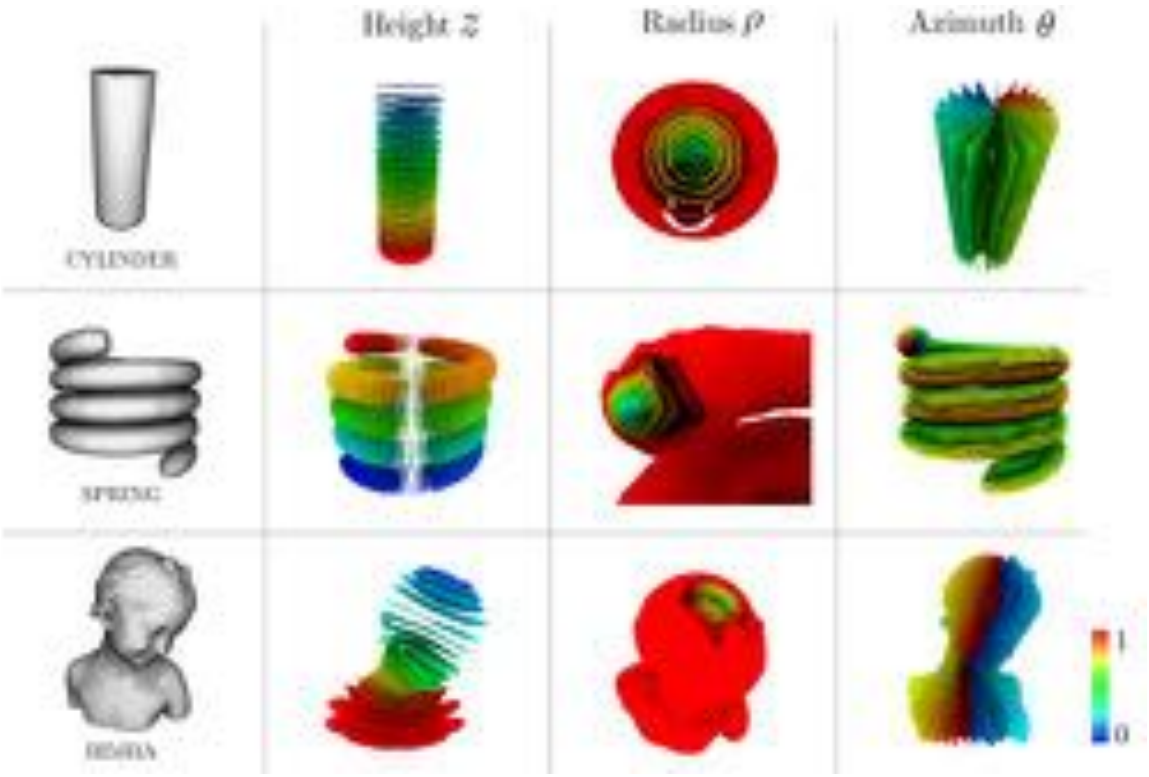


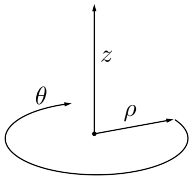
Figure 1: Three different examples of our volumetric cylindrical parameterization. For each map, we show a sub-sampling of the iso-surfaces of each scalar field (z, ρ, θ). Our approach is capable of producing high quality volumetric maps for simple cylinders with straight axis (top); it does not introduce twisting of the azimuth field along the spine for cylinders with a tortuous axis (middle); and it is capable of producing high quality fields also for shapes with weak cylindricity (bottom).

for tubular shapes. Some of them aim to produce block-structured surface meshes [35, 12, 32, 36], some others volumetric meshes [37, 6, 38, 39]. These methods focus on a specific tessellation. We offer a generic tool for solid modeling based on a cylindrical map and are therefore more general and open to various domain discretization choices.

3. Preliminaries and Notation

In this section, we briefly introduce the two fundamental bricks on top of which we define our algorithm: cylindrical polar coordinates and cylindrical parameterization. In doing so, we also fix the notation, which will be consistent in all the subsequent sections.

Cylindrical Coordinates. extend the two-dimensional polar coordinates to three dimensions by superposing a height axis. The position of a point in cylindrical space is defined by the radial distance from the axis, called **radius**, the angular distance with respect to a reference direction, called **azimuth**, and the distance from a reference plane orthogonal to the axis, called **height**. Different symbols have been used to represent these three quantities — we adopt here the notation proposed in [40], using ρ to denote the radius, θ to denote the azimuth, and z to denote the height.



Cylindrical Parameterization. is a bijection between a tubular shape $\mathcal{T}$ composed of two disjoint bases and a lateral surface in between and a unit cylinder living in the cylindrical coordinate reference system. We are interested in the *volumetric* version of this map, meaning that to each point in the interior of $\mathcal{T}$ we associate a triplet (ρ, θ, z) that maps it to a point in the cylindrical parametric space, with $\rho \in [0, 1]$, $\theta \in [0, 2\pi)$ and $z \in [0, 1]$.

Notice that the cylindrical parameterization is singular along the axis, where the radius vanishes. If we fix a height z, for any pair of azimuth values θ_1, θ_2 we have that $(0, \theta_1, z) \equiv (0, \theta_2, z)$. Similarly, if we fix a radius ρ and a height z, we have that $(\rho, 0, z) \equiv (\rho, 2\pi, z)$. This is because the azimuth component of the map is also periodic in $[0, 2\pi)$. The structure of the parametric space is particularly relevant when it comes to compute and store a cylindrical map in the vertices of a discrete mesh. In fact, the domain discretization must account for the periodicity and singularity of the polar map. Generating such a discretization will be the subject of Section 5.

4. Overview

We propose an automatic method to generate solid cylindrical maps of complex tubular shapes with arbitrary topology. Our inputs are the boundary of a tubular object $\mathcal{T}$, represented by a triangle mesh, and its curve-skeleton $\mathcal{S}$ (in the sense of [20]). Assuming $\mathcal{S}$ is composed of n bones, we first decompose $\mathcal{T}$ into a set of atomic tubes $\mathcal{T}^1, \mathcal{T}^2, ..., \mathcal{T}^n$ (each one associated to a bone of $\mathcal{S}$), then we generate a separate map for each of them (Section 6).

In the discrete setting, the map is defined on the vertices of a tetrahedral mesh representing $\mathcal{T}$, and linearly interpolated in the interior of each tetrahedron. In particular, the map is represented by three discrete harmonic functions (f_ρ, f_θ, f_z) that associate respectively a radius, an azimuth and a height to each vertex of such mesh.

In Section 5, we detail how to parameterize an atomic tubular component. We tackle the problem of generating a discretization of the shape which takes into account the periodic and singular structure of the parametric domain, and also discuss how to compute the fields f_ρ, f_θ, f_z and encode them in the vertices of the resulting tetrahedral mesh. To the best of our knowledge, this is the first method capable to produce such a discretization.

In Section 6, we discuss our skeleton-driven decomposition in atomic tubes. We motivate the introduction of a novel splitting technique by observing that similar methods for tube decomposition (e.g. [30, 31]) may leave portions of the object with weak cilindricity outside of the decomposition and are therefore not suited for our purposes. The decomposition we present guarantees that the union of all tubes coincides with the input shape, thus ensuring that the resulting parameterization is defined within the whole domain.

5. Single Tube Parameterization

Herewith, we consider an atomic tubular shape $\mathcal{T}$ defined by sweeping an evolving disk-like, possibly non-planar cross section along a general curve in the space. We call such curve the **axis** of the tube, and we denote it with $\mathcal{T}_{axis}$. We refer to the uppermost and lowermost cross sections of $\mathcal{T}$ as its **bases**, and we call them $\mathcal{T}_{top}$ and $\mathcal{T}_{bot}$ respectively. Finally, we refer to the portion of the boundary in between $\mathcal{T}_{top}$ and $\mathcal{T}_{bot}$ as the **lateral surface** of the tube, and we call it $\mathcal{T}_{srf}$. Notice that the union of $\mathcal{T}_{top}, \mathcal{T}_{bot}$ and $\mathcal{T}_{srf}$ coincides with the boundary of $\mathcal{T}$.

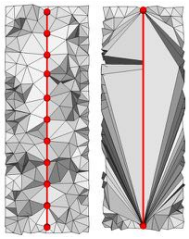
We first explain how to generate an initial volumetric discretization of the tube (Section 5.1), and then discuss the details regarding the computation of the radius (Section 5.2), height (Section 5.3), and azimuth (Section 5.4) of our cylindrical map.

5.1. Pre-processing

The goal of this step is to prepare the volumetric mesh on top of which the functions f_ρ and f_z will be computed. The domain will then be further modified to account for the periodic structure of f_θ, as explained in Section 5.4. In the discrete setting, $\mathcal{T}$ is a solid bounded by a triangle mesh and $\mathcal{T}_{axis}$ is a piece-wise linear open curve defined by a collection of segments attached to one another at their extremities. We initialize $\mathcal{T}_{axis}$ as the curve-skeleton of $\mathcal{T}$. Our method is oblivious to the specific method being used. $\mathcal{T}_{axis}$ can be either computed with an algorithm (e.g., [41]) or sketched by a user.

We wish to generate a tetrahedral mesh that satisfies two important requirements: (i) the radius component f_ρ evaluates 0 at the tube axis ($\mathcal{T}_{axis}$) and 1 at its lateral surface ($\mathcal{T}_{srf}$). For this to be possible, both the vertices and the edges that compose $\mathcal{T}_{axis}$ should be incorporated in the tessellation; (ii) the height component f_z associates a disk-like cross section to each point in $\mathcal{T}_{axis}$, with the uppermost and lowermost cross-sections being the bases of the tube. The tube axis $\mathcal{T}_{axis}$ should therefore traverse $\mathcal{T}$ from base to base and be completely internal to it, with the only exception being its two endpoints, which must be placed directly on its surface.

Unfortunately, the axis computed by some algorithms [31] is entirely contained in the solid, endpoints included. In these cases, we simply elongate each of its extremities by linear extrapolation, locate the intersection with the boundary of $\mathcal{T}$, and create an additional segment for each extremity of $\mathcal{T}_{axis}$. Whenever necessary, we split the involved triangles to ensure mesh conformity. Conformity with respect to a boundary mesh and a collection of segments are easy requirements for many tetrahedral meshing algorithms. We eventually generate the tetrahedral mesh using off-the-shelf software [42]. Notice that since $\mathcal{T}_{axis}$ will be embedded in the mesh, it is important to have a good sampling of the curve. A coarse sampling will result in a bad volumetric discretization, as shown in the inset aside. If necessary, then $\mathcal{T}_{axis}$ can be equally resampled via arc-length parameterization before meshing.



5.2. Radius

The function f_ρ that encodes the radius of the cylindrical parameterization is the simplest to compute. It evaluates 0 at the tube axis ($\mathcal{T}_{axis}$), and 1 at its lateral surface ($\mathcal{T}_{srf}$). We generate it by solving for the following harmonic field

$$\Delta f_\rho = 0 \,{}_{\begin{smallmatrix}|_{\mathcal{T}_{axis}} = 0\\ |_{\mathcal{T}_{srf}} = 1\end{smallmatrix}} \quad .$$

We implement Δ as the cotangent Laplace operator for tetrahedral meshes

$$\Delta(v_i) = \sum_{v_j \in N_i} \left(\frac{1}{6} \sum_{t \in N_{(i,j)}} |t_{(p,q)}| \cot \varphi_{t_{(p,q)}} \right) (v_i - v_j),$$

where v_i is a general vertex of the tetrahedral mesh, N_i its directly adjacent vertices, $N_{(i,j)}$ is the fan of tetrahedra having (i,j) as edge and, for each such tetrahedron, $t_{(p,q)}$ is the edge (p,q) opposite to (i,j), with $|t_{(p,q)}|$ and $\varphi_{t_{(p,q)}}$ being its length and dihedral angle, respectively.

5.3. Height

The height of the cylindrical parameterization (f_z) evaluates 0 at the bottom base of the tube ($\mathcal{T}_{bot}$), and 1 at its top base ($\mathcal{T}_{top}$). In order to be able to compute it, it is therefore necessary to select two disjoint regions on the boundary of $\mathcal{T}$ that act as bases. Each base is attached to one of the extrema of the tube axis, therefore we start from the endpoints of $\mathcal{T}_{axis}$ and we perform a BFS visit of the graph having as nodes the boundary vertices of $\mathcal{T}$ and as arcs its surface edges. Starting from each endpoint we expand by vertex normal similarity, according to the following criterion

$$\vec{n}_v \cdot \vec{n}_{ref} > 0.9,$$

where $\vec{n}_v$ is the surface normal at the current vertex and $\vec{n}_{ref}$ is the surface normal at the axis endpoint currently considered. Notice that if the result of this procedure is not satisfactory manual refinement of the bases is also possible. In our experiments we always used the full automatic procedure, at the end of which we solved for the height field

$$\Delta f_z = 0 \Big|_{\substack{\mathcal{T}_{bot} = 0 \\ \mathcal{T}_{top} = 1}} .$$

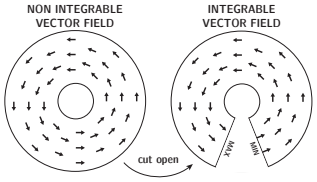
using the two bases $\mathcal{T}_{top}$ and $\mathcal{T}_{bot}$ as Dirichlet boundary conditions.

5.4. Azimuth

The function f_θ that encodes the azimuth of the cylindrical map is a periodic function defined in the range $[0, 2\pi)$. A desirable property for f_θ is to have a gradient that is locally as-orthogonal-as-possible to both ∇f_z and ∇f_ρ. This is a fundamental requirement for applications like hexahedral re-meshing and spline fitting, as it ensures well shaped hexahedra and tensor product trivariate domains [16, 11]. Assuming to have already computed both height (f_z) and radius (f_ρ), the optimal gradient ∇f_θ can be trivially computed on a per-tet basis as

$$\nabla f_\theta = \nabla f_z \times \nabla f_\rho.$$

Unfortunately, this relation is not of much help because the resulting vector field is divergence-free. Vector fields defined on the parameterization domain (see figure below) with divergence zero everywhere do not have a scalar potential, meaning that they are not the gradient of any scalar function and, therefore, are not integrable [43]. In order to be able to compute f_θ and encode its values in a discrete mesh, it is therefore necessary to disambiguate the singular and periodic portions of the domain (Section 3), duplicating the vertices with azimuth 0 and 2π, and removing the singular axis by carving the shape along its spine, as depicted in the 2D example above.



We compute a surface Ψ that cuts the cylinder open along its axis (Section 5.4.1) and produce a new tetrahedral mesh that conforms to both $\mathcal{T}$ and Ψ. We then duplicate the vertices and faces of Ψ in order to generate two copies of it, say Ψ_{front} and Ψ_{back}. For the singularity along the spine, we create a cavity that follows the tube's axis. To do so, we consider the radius field f_ρ and compute the iso-surface $\rho = \epsilon$ (0.05, in all our tests). We then split all the tetrahedra crossed by the iso-surface and eliminate the tets staying in the $\rho < \epsilon$ side of the mesh. We eventually compute the function f_θ in the resulting tetrahedral mesh, by solving for the following harmonic field subject to Dirichlet boundary conditions on both sides of Ψ

$$\Delta f_\theta = 0 \Big|_{\substack{\Psi_{front} = 0 \\ \Psi_{back} = 2\pi}} .$$

5.4.1. Computation of the cut surface

The design of Ψ is a delicate part in the generation of f_θ. Its shape, indeed, influences the distortion of the resulting parameterization. As already explained in the beginning of the section, we would like f_θ to be as-orthogonal-as-possible to both f_z and f_ρ. Since the field f_θ will have Ψ_{front} as source and Ψ_{back} as sink, the cut surface must be as-aligned-as-possible to the gradients ∇f_z and ∇f_ρ. This property guarantees well shaped cuts, avoiding tedious artifacts such as twisting along the tube axis $\mathcal{T}_{axis}$.

In short, we define Ψ as a topological quad $(\psi_0, \psi_1, \psi_2, \psi_3)$ in the parametric space, as depicted in the right part of Figure 2. The parametric coordinates of each corner are

$$\begin{aligned} f_z(\psi_0) &= 0, \quad f_\rho(\psi_0) = 1 \\ f_z(\psi_1) &= 1, \quad f_\rho(\psi_1) = 1 \\ f_z(\psi_2) &= 0, \quad f_\rho(\psi_2) = 0 \\ f_z(\psi_3) &= 1, \quad f_\rho(\psi_3) = 0. \end{aligned}$$

Notice that ψ_2 and ψ_3 are the endpoints of the $\mathcal{T}_{axis}$.

To construct the outline of Ψ, we start at an arbitrary point ψ_0 in the boundary of the bottom base $\mathcal{T}_{bot}$ and we trace a curve over its lateral surface, following the gradient ∇f_z until we reach its top base (Figure 2, left). The endpoint of the resulting curve is ψ_1. We then trace a new curve over the bottom base of the tube, starting from ψ_0 and following the anti gradient $-\nabla f_\rho$ until we reach ψ_2. We do the same on the top base, starting form ψ_1 and following $-\nabla f_\rho$ until we reach the point ψ_3. These three curves, together with the axis $\mathcal{T}_{axis}$, form the outline of the topological quad Ψ (Figure 2, middle). We integrate the integral curves in the connectivity by splitting all the edges and triangles being traversed.

To design the interior of Ψ, we evenly sample the curve connecting ψ_0 and ψ_1 and, starting from each such sample,

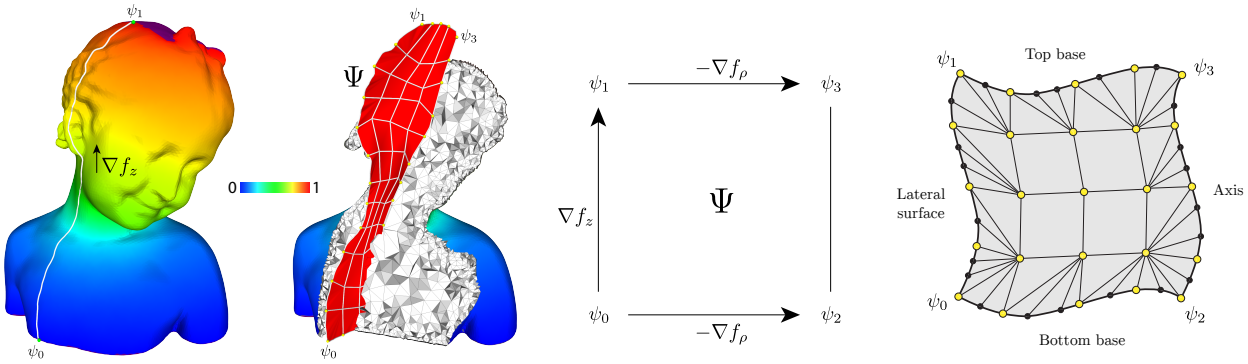


Figure 2: We cut the tubular domain open to accommodate a cylindrical map. We start from an arbitrary point at its base and we trace an integral curve along its lateral surface, following the gradient ∇f_z (left). We then sub-sample the curve and, for each such sample, we create an integral curve that penetrates the volume along $-\nabla f_\rho$. The result is a gridded topological quad that defines our cut surface Ψ (middle). We mesh the grid using quads for the interior and triangle fans for the cells with at least one side exposed either on the boundary of the tube or on its inner axis. This ensures a seamless integration of Ψ in the discrete domain, preserving mesh conformity everywhere (right).

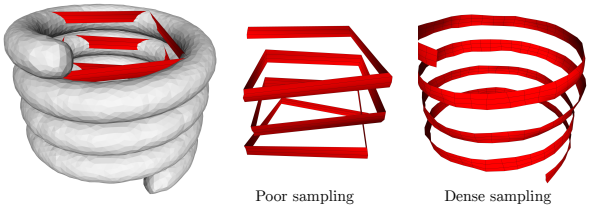


Figure 3: If the sampling density is not dense enough, then the cut surface Ψ can be outside of the cylinder (left and middle). Increasing the sampling frequency fixes the problem (right). A sufficiently dense sampling, such that Ψ is fully contained in the cylinder and conforming with its boundary, is guaranteed to exist.

we trace an integral curve that penetrates the volume of the cylinder along the anti gradient $-\nabla f_\rho$. Since $\mathcal{T}_{axis}$ is the minimum of the radius field f_ρ, each such curve will converge to it. We eventually re-sample each curve in order to generate a regular sample grid that we use to drive the meshing process.

Since the four sides of Ψ are exposed either on the boundary of the tube or on its inner axis, we mesh the sample grid using quads for the interior and triangle fans for the cells. This ensures that the lateral surface $\mathcal{T}_{srf}$, the bases $\mathcal{T}_{top}, \mathcal{T}_{bot}$, the axis $\mathcal{T}_{axis}$ and Ψ are all seamlessly integrated in a single conforming mesh (Figure 2, right). The ultimate tetrahedral mesh is eventually constructed via refinement, that is, splitting all the tets traversed by Ψ. Notice that mesh conformity is fundamental here. Self-intersecting triangles will not be tolerated by TetGen [42] causing a failure in the refinement, whereas small gaps between Ψ and $\mathcal{T}$ will put the two sides of the cut surface directly in contact, making it useless.

This approach has a flaw: the triangle fans used to mesh the borders of Ψ may self intersect or even sneak out of the domain if the grid sampling is not dense enough, as shown in Figure 3. However, since both the height and the radius are harmonic functions, a sampling density $\tau > 0$ for which the surface is both fully contained in the tube and does not self intersect is guaranteed to exist. In our implementation, the user can visually inspect the cut surface and, if necessary, increase the sampling density both along f_z and f_ρ. An automatic detection of pathological situations is also possible. Testing the cut surface and the boundary of the tube for self-intersections would highlight the spots that need a denser sampling.

6. Skeleton-driven tube decomposition

The method described so far is capable of generating a cylindrical map for simple tubular shapes of trivial topology. Although mappings of this kind can already be useful for some specific tasks [13], we acknowledge that in most applications shapes will be much more complex, containing an arbitrary number of tubes possibly connected together at high valence junctions. In this section, we explain how to generalize the cylindrical map to any shape that can be well described by a curve-skeleton, regardless the number and connectivity of the bones.

Given a complex tubular shape $\mathcal{T}$ and its curve-skeleton $\mathcal{S}$, we introduce a method to decompose $\mathcal{T}$ into a set of atomic tubes $\mathcal{T}^1, \mathcal{T}^2, ..., \mathcal{T}^n$, each associated with a bone of $\mathcal{S}$. Ideally, we aim to assign each point in the object to its closest skeleton bone, which corresponds to partitioning the domain by a Voronoi diagram having the bones themselves as sites. Computing a Voronoi diagram of 3D line segments is known to be an open problem in literature, but approximate solutions exist [44]. However, we observe that a strict Voronoi decomposition would possibly lead to undesired results, where some of the elements of the decomposition do not have a clear tubular structure (Figure 4).

Structural constraints. Our tubular decomposition tries to be as close as possible to a Voronoi partitioning, while satisfying precise structural requirements imposed by the underlying curve-skeleton. We make sure that each element of the decomposition is a tube-like shape composed of a lateral surface bounded by two bases, and is traversed from

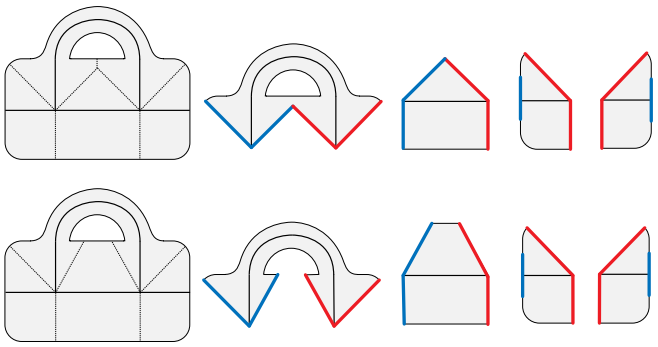


Figure 4: Top: a Voronoi decomposition of the domain which assigns each point to its closest skeleton bone may fail to generate a valid tubular decomposition (first and second components). Bottom: our method tries to be as Voronoi as possible while satisfying precise structural requirements that ensure that each component is a topological tube, with two disjoint bases (red and blue) and a skeleton bone as inner axis (black).

base to base by a skeleton bone which represents its inner axis. The lateral surface must be completely exposed on the boundary of the domain, whereas the bases are either skinned into the volume of the shape (for skeleton joints where at least two bones meet) or produced as described in Section 5.3 (for the extremities). These constraints ensure that tubes touch to one another only at their bases, and are completely disjoint if their corresponding bones are not directly connected through a skeleton branching node. Notice that the decomposition we aim to find is inherently volumetric, as finding the tube bases skinned into the volume is part of the problem. For this and for other reasons, none of the previous surface-based decomposition approaches (e.g., [30, 31]) is capable of solving this problem (Section 2).

Skeleton split. Internal skeleton bones have two branching nodes as extrema. In order to separately control the connectivity between adjacent tubes on each side of the bone, we halve each curve of the skeleton. As a result, each half-bone will be adjacent to its twin half-bone on one side, and to the half bones incident to the same branching node on the other side.

Problem formulation. We cast the half-tube decomposition as a multi-label graph optimization problem defined on the dual of the tetrahedral mesh representing the domain $\mathcal{T}$. The graph has a node for each tetrahedron and an arc for each pair of adjacent tetrahedra connected through a facet. Our decomposition is the minimizer of the following energy

$$E(\mathcal{T}) = \sum_{t} C(t,b) + \sum_{t_i,t_j} S(t_i,b_i,t_j,b_j),$$

where $C(t,b)$ is the so called *data term*, that is, the cost of assigning the tetrahedron t to the skeleton half-bone $\mathcal{S}^b$, and $S(t_i,b_i,t_j,b_j)$ is the *smoothing term*, that is, the cost of assigning the pair of adjacent tetrahedra t_i, t_j to the skeleton half-bones $\mathcal{S}^{b_i}, \mathcal{S}^{b_j}$, respectively.

The data term is defined in the range $[0,1]$ and is computed as the distance between the barycenter of the tetrahedron $\bar{t}$ and the skeleton half-bone $\mathcal{S}^b$, normalized by the maximum distance between a tet and a skeleton half-bone. If t is directly incident to $\mathcal{S}^b$ (i.e., they share one edge or vertex), then $C(t,b)$ evaluates zero

$$C(t,b) = \begin{cases} 0 & \text{if } \mathcal{S}^b \cap t \neq \emptyset, \\ \frac{d(\bar{t},\mathcal{S}^b)}{\max_{t,b} d(\bar{t},\mathcal{S}^b)} & \text{otherwise.} \end{cases}$$

The smoothing term is defined in the range $[0,\infty)$ and evaluates 0 if two adjacent tetrahedra map to the same skeleton half-bone, 1 if they map to two adjacent half-bones (i.e. the half-bones meet at a skeleton branching node or are part of the same bone), and ∞ if they map to two disjoint skeleton half-bones

$$S(t_i,b_i,t_j,b_j) = \begin{cases} 0 & \text{if } \mathcal{S}^{b_i} = \mathcal{S}^{b_j}, \\ 1 & \text{if } \mathcal{S}^{b_i} \cap \mathcal{S}^{b_j} \neq \emptyset, \\ \infty & \text{if } \mathcal{S}^{b_i} \cap \mathcal{S}^{b_j} = \emptyset. \end{cases}$$

We minimize the energy $E(\mathcal{T})$ using the graph cut algorithm [45, 46, 47] and generate the ultimate tube decomposition by merging together the regions associated to twins half-bones. Notice that any tube decomposition for which $E(\mathcal{T}) < \infty$ is *valid*, meaning that it satisfies the structural constraints imposed by the underlying curve-skeleton. Since we are solving the decomposition problem in a discrete space, a solution for which $E(\mathcal{T}) < \infty$ may not exist. If the mesh is sufficiently dense, then this case is very unlikely to happen (it never happened in all our experiments). If necessary, then additional degrees of freedom may be introduced by mesh refinement, up to a point at which a valid solution is guaranteed to exist. With a tube decomposition at hand, we first separate tubes by duplicating the vertices shared between multiple components, and then generate a cylindrical map for each component as explained in Section 5.

7. Continuity of the map

We discuss here the continuity of the maps both for the single and multiple tube cases. As for similar approaches based on discrete harmonic functions, for the single tube case the map is $\mathcal{C}^1$ continuous everywhere except at the boundary conditions, where it is $\mathcal{C}^0$ continuous. For the multiple tube case, similarly to cuboidal maps [16] the goal is to ensure $\mathcal{C}^0$ across the boundaries between adjacent tubes. In the following paragraphs, we discuss each coordinate separately.

Radius. The interfaces between adjacent tubes are conforming. Consequently, when solving for f_ρ it is possible to impose $\mathcal{C}^0$ continuity across the tubes by simply asking each copy of these interface vertices to receive the same

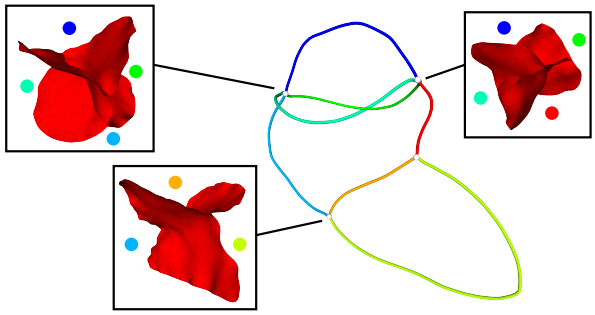


Figure 5: We show here the membranes that separate adjacent tubes for the Fertility model. All the vertices in these membranes will have multiple parametric coordinates, one for each tube incident to them. We impose equality constraints on the height and radius components of these parametric coordinates. Doing so, we can ensure global $\mathcal{C}^0$ continuity for both height and radius.

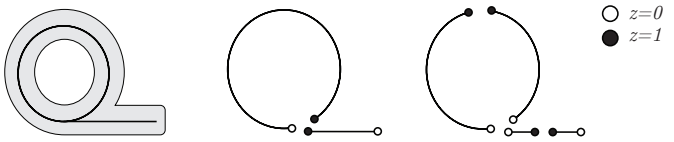


Figure 6: If the skeleton contains a loop (left) it might not be possible to generate a $\mathcal{C}^0$ continuous f_z field (middle). By virtually splitting each bone into two sub-bones we can guarantee that tubes always touch at their lower base ($z = 0$), thus ensuring $\mathcal{C}^0$ continuity everywhere (right).

ρ value in all the tubes incident it (Figure 5). This translates to simple linear constraints that can be added to the problem described in Section 5.2 as additional Dirichlet boundary conditions.

Height. As depicted in Figure 6 (middle), in some cases it is not possible to have f_z being $\mathcal{C}^0$ continuous across adjacent tubes. However, continuity can be always ensured by virtually halving each tube, producing a new function $\tilde{f}_z$ that evaluates 0 at both bases and 1 at mid height (Figure 6, right). We defined $\tilde{f}_z$ as

$$\tilde{f}_z = \begin{cases} 2f_z & 0 \leq f_z \leq 0.5, \\ -2(f_z - 1) & 0.5 < f_z \leq 1. \end{cases}$$

Azimuth. As for the height case, it is not possible to ensure continuity of the azimuth for three or more tubes that meet at a branching node. We decided not to ensure any continuity in the f_θ fields coming from adjacent tubes, but rather store for each pair of tubes an angular displacement $\Delta\theta$ that, if necessary, allows to match the θ fields pairwise. Notice that for many applications (e.g. hexahedral and shell meshing) this is not limiting. In fact most meshing approaches use templates to handle the branching nodes [35, 39] and do not require to evaluate the parameterization at the boundaries between adjacent tubes. Nonetheless, we believe that trying to ensure some sort of continuity of the azimuth fields across more than two adjacent tubes is an interesting problem, and we plan to investigate this topic further in future works.

Model	min	avg	max
Cylinder	1.101	3.453	19.718
Spring	1.071	3.537	18.147
Bimba	1.117	8.312	22.908
Femur (1 bone)	1.107	6.358	25.734
Cactus	1.185	3.649	21.676
Rocker Arm	1.201	7.961	23.819
Fertility	1.134	5.164	30.827

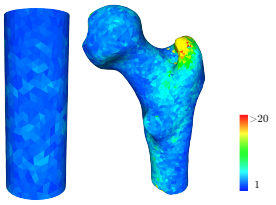


Figure 7: Left: we report some statistics about the min/avg/max distortion of the cylindrical maps shown throughout the paper. We use the aspect-ratio distortion metric [15] — the optimal value (i.e., no distortion) is 1. Mappings are provided as-is, without any post processing. Notice that lower distortion maps could be easily obtained by applying some optimization technique, such as [48, 49, 15]. Right: as expected a cylinder has a fairly low distortion whereas shapes with weak cylindricity, such as the Femur, present areas with higher distortion around protuberances.

8. Results

We implemented a C++ prototype of our solid cylindrical map on a MacBook Pro equipped with an Intel Core i5 and 16GB of RAM. We used [50, 41, 51, 52] to compute the curve-skeletons and [53] for numerics. In Table 1, we list numeric results for all our models. Due to the explicit encoding of the map, our pipeline runs one order of magnitude faster than [13], it is more efficient to evaluate and does not require customized code for standard operations such as iso-contouring. In Figure 7 we discuss the distortion induced by our cylindrical map. Notice that maps are provided as-is, without any post-processing. Aspect-ratio distortion could be significantly lowered by applying some optimization technique for simplicial maps, such as [48, 49, 15].

Single tube. In Figure 1, we show three examples of our solid cylindrical map for single tubular components. As can be noticed, our method does not introduce twisting of the azimuth field along the spine for cylinders with a tortuous axis, such as the Spring model, and it is capable of producing high quality fields also for shapes with weak cylindricity, such as the Bimba model. For each data set, we show a color coded sub sampling of the of the iso-surfaces of radius, azimuth and height. Differently from [13] our map is fully explicit, therefore iso-contouring can be trivially performed with any available implementation of marching cubes for unstructured meshes.

Multiple tubes. In Figure 8, we show a gallery of results for tubular models with arbitrary topology. In the first column, we show the result of the skeleton-driven decomposition described in Section 6. For each tube, we then generate a separate cylindrical map. Our method applies to different classes of shapes, ranging from simple tubular objects such as the Cactus, tubular objects with complex

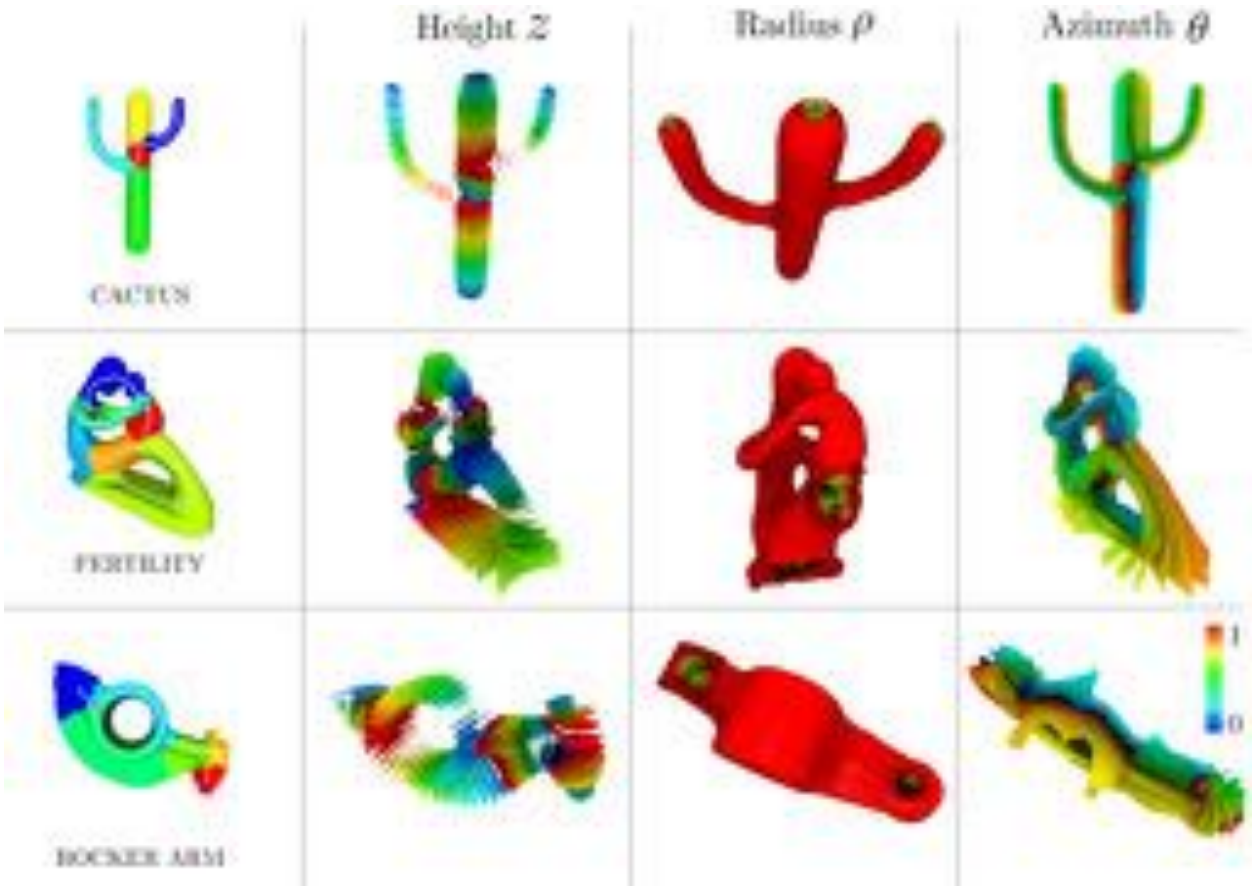


Figure 8: We extend solid cylindrical maps to arbitrary shapes by splitting each shape into a set of topological tubes (left column) and computing a separate cylindrical map for each component (second, third and forth columns). Our method applies to simple tubular shapes with trivial topology such as the Cactus (top), complex tubular shapes such as Fertility (middle), but also to shapes with weak cylindricity, such as the RockerArm model (bottom).

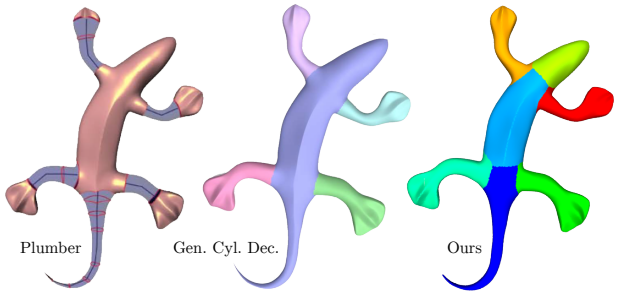


Figure 9: A comparison between Plumber [31], GCD [30] and our tube decomposition. Both Plumber and GCD do not decompose the core of the Gecko, leaving a spurious component with four disjoint boundary curves. It is unclear how to fit a cylindrical map in there. Our method decomposes the core of the shape into three neat tubular parts (head, tail, body) which admit a natural cylindrical map.

topology such as Fertility, but also composite objects with weak cylindricity, such as the Rocker Arm.

Comparison with Plumber and GCD. In Figure 9, we compare our decomposition with Plumber [31] and the Generalized Cylinder Decomposition [30]. As can be noticed, the

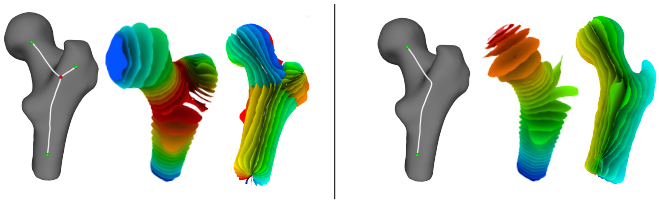


Figure 10: By editing the underlying curve-skeleton, we can control the cylindrical map. Here, we show two alternative cylindrical maps for the Femur, obtained by considering a skeleton with three (left) and one (right) bones respectively. For each example, the height and azimuth components of the map are shown.

decompositions provided by both these techniques contain spurious elements which do not have a clear tubular structure. Our method produces a complete decomposition in topological tubes. Furthermore, being volumetric, it also provides the separation interfaces between adjacent tubes as a by product. This information is not available in the other techniques we considered.

Skeleton control. The interplay between curve-skeletons and cylindrical maps can be exploited for interactive mod-

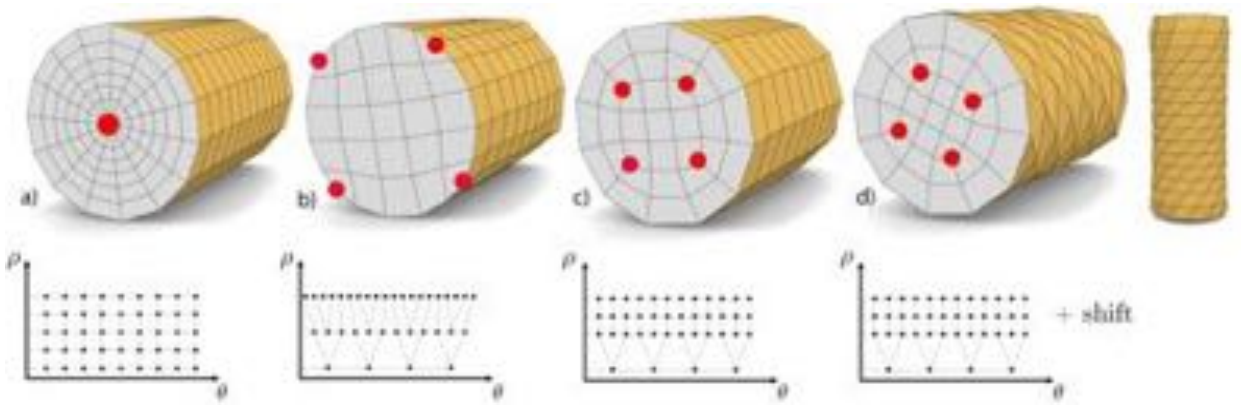


Figure 11: We sample the cylindrical parametric space in order to generate four hexahedral meshes with different connectivity: polar (a); grid-like (b) and mixed (c,d). In the bottom line, we show the sampling pattern on the $\theta\rho$ plane. The volumetric mesh is eventually generated by extruding the same pattern along the z axis. For the torsion example (d), at each layer we shift the pattern along the θ axis — this operation corresponds to a rotation in the shape space. Red dots represent the singular vertices of the mesh. As the sampling strategy changes, singularities move from the core (a) to the surface (b) to the middle (c,d) of the volume. This mechanism allows users to finely control singularity placement, one of the most critical aspects of mesh generation.

eling. In Figure 10, we show two different cylindrical maps for the Femur data set, obtained by editing the underlying curve-skeleton. Indeed, the curve-skeleton becomes a tool to control the solid map. A number of intuitive interfaces for skeleton editing exist in literature, both academic [21] and commercial. We believe that editing the position of the skeleton points, or adding/removing bones from the skeleton is a nice and intuitive way to control a solid map. As future work, we plan to exploit this concept further, and propose new applications based on this paradigm.

Hexahedral meshing. One of the applications we target is the generation of hexahedral meshes, which can be refined for FEM analysis or used as control cages for spline fitting. The solid map we provide is a powerful tool that enables for a number of different domain discretization choices that correspond to different sampling patterns of the cylindrical parametric space. One of the most difficult aspects in mesh generation is to control the singularity placement. Using a cylindrical map singularities can be pushed towards the interior or the boundary, or placed at an angular distance w.r.t. the inner axis simply by translating the meshing pattern along the ρ and θ axis, respectively (Figure 11). For objects composed of multiple tubes, our meshing strategy can be complemented by techniques such as [35, 39], which propose neat methods to define cuboidal structures for junctions with arbitrary topology.

Shell Meshing. Our solid map can also be used for the generation of shell meshes and nested meshes for multi-grid solvers [54]. In Figure 12, we show a simple example for the Femur data set, composed of three nested shell meshes completely disjoint to one anther. We obtained such meshes by filtering the radius component of the map. At the bottom of the figure, we show the color-coded sampling patterns we used.

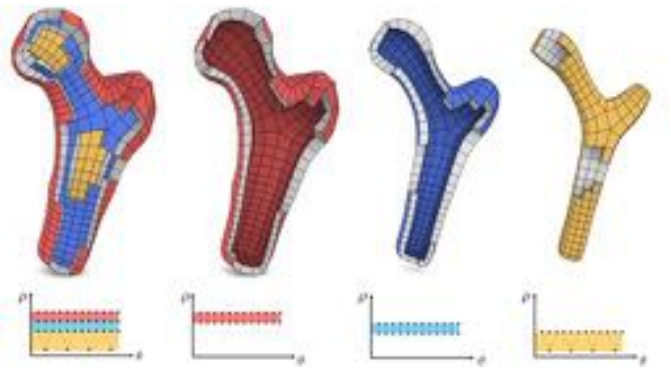


Figure 12: Shell meshes can be easily generated by filtering the radius parameter ρ of our cylindrical map. Here, we show a shell mesh composed of three nested layers of the Femur data set. For the outer mesh (red) we sampled the portion of parametric space between $\rho = 1$ and $\rho = 0.8$; for the middle mesh (blue) we sampled the space between $\rho = 0.8$ and $\rho = 0.6$; for the core mesh (orange) we sampled the parameter space with $\rho \leq 0.6$. The three meshes are topologically disjoint to one another.

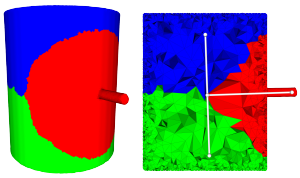
9. Conclusions

In this work, we extended solid cylindrical maps to any general tubular shape, with no restrictions on its topology and morphology. Differently from previous approaches, our maps are *explicit*, meaning that the parametric coordinates are directly encoded in the vertices of a discrete mesh, and the map can be efficiently evaluated in both directions and exploited by off-the-shelf algorithms. As a direct consequence our map is much more efficient to evaluate and does not require customized code to implement standard operations such as iso-contouring. The generation of a discrete domain that accounts for the singular and periodic structure of the parametric space is also part of our contribution.

Model	tris	tets	bones	t_{split}	t_z	t_ρ	t_θ
Cylinder	2K	19K	1	-	0.09	0.03	0.07
Spring	15K	53K	1	-	1.38	0.40	0.76
Bimba	46K	190K	1	-	17.95	12.96	14.11
Femur	8K	39K	1	-	4.16	1.99	2.98
Femur	8K	50K	3	0.99	1.95	1.16	1.59
Cactus	10K	52K	5	1.02	1.09	0.62	0.84
R.-Arm	28K	115K	6	3.41	4.06	2.20	3.30
Fertility	34K	91K	7	3.15	8.84	4.62	7.03

Table 1: For each model, we show: number of triangles of the input mesh (tris); number of tets of the output mesh (tets); number of bones of the curve-skeleton (bones) and times (in seconds) for each step of the algorithm, that is, tube decomposition (t_{split}), computation of height (t_z), radius (t_ρ) and azimuth (t_θ).

Limitations and future work. We observe that shapes with inner cavities (e.g., a hollow sphere) do not admit a homotopic curve skeleton, hence cannot be processed by our pipeline. Although in principle it works on any other shape that admits such a skeleton, our method is intended for tubular shapes and may lead to non intuitive or excessively distorted maps for shapes which do not have a clear tubular structure. We emphasize that the skeleton-driven decomposition presented in Section 6 is purely topological and, as such, does not strive to optimize the geometry of the cut or to align to sharp features (e.g. for CAD models). Consequently, when very thick and very thin tubes meet at a junction point poor (though still valid) decompositions may arise, as depicted in the inset aside. We leave the quest for good geometric cuts as a future work. A promising approach successfully exploited in [3] could be to enrich the smoothing component of our energy term with geometric measures, such as dihedral angles. Finally, as explained in Section 7 our map lacks $\mathcal{C}^0$ continuity across adjacent tubes in the azimuth component. We plan to tackle this problem in future work.



References

[1] M. Spagnuolo, Shape 4.0: 3D shape modeling and processing using semantics, IEEE Computer Graphics and Applications 36 (1) (2016) 92–96.

[2] X. Li, X. Guo, H. Wang, Y. He, X. Gu, H. Qin, Harmonic volumetric mapping for solid modeling applications, in: Proc. of Symposium on Solid and Physical Modeling, 2007, pp. 109–120.

[3] M. Livesu, N. Vining, A. Sheffer, J. Gregson, R. Scateni, Polycut: Monotone graph-cuts for polycube base-complex construction, ACM Trans. Graph. 32 (6).

[4] D. Sokolov, N. Ray, L. Untereiner, B. Lévy, Hexahedral-dominant meshing, ACM Trans. Graph. 35 (5) (2016) 157.

[5] K. Hu, Y. J. Zhang, T. Liao, Surface segmentation for polycube construction based on generalized centroidal voronoi tessellation, Computer Methods in Applied Mechanics and Engineering.

[6] L. Liu, Y. Zhang, Y. Liu, W. Wang, Feature-preserving t-mesh construction using skeleton-based polycubes, Computer-Aided Design 58 (2015) 162–172.

[7] B. Li, X. Li, K. Wang, H. Qin, Surface mesh to volumetric spline conversion with generalized polycubes, IEEE Trans. on Visualization and Computer Graphics 19 (9) (2013) 1539–1551.

[8] H. Xu, W. Yu, S. Gu, X. Li, Biharmonic volumetric mapping using fundamental solutions, Visualization and Computer Graphics, IEEE Trans. on 19 (5) (2013) 787–798.

[9] T. Martin, E. Cohen, Volumetric parameterization of complex objects by respecting multiple materials, Computers & Graphics 34 (3) (2010) 187–197.

[10] Y. Wang, X. Gu, S.-T. Yau, et al., Volumetric harmonic map, Communications in Information & Systems 3 (3) (2003) 191–202.

[11] X. Gao, T. Martin, S. Deng, E. Cohen, Z. Deng, G. Chen, Structured volume decomposition via generalized sweeping, IEEE Trans. on Visualization and Computer Graphics 22 (7).

[12] J. A. Bærentzen, R. Abdrashitov, K. Singh, Interactive shape modeling using a skeleton-mesh co-representation, ACM Trans. Graph. 33 (4) (2014) 132.

[13] T. Martin, E. Cohen, M. Kirby, Volumetric parameterization and trivariate b-spline fitting using harmonic functions, in: Symposium on Solid and Physical Modeling, ACM, 2008, pp. 269–280.

[14] G. M. Treece, R. W. Prager, A. H. Gee, Regularised marching tetrahedra: improved iso-surface extraction, Computers & Graphics 23 (4) (1999) 583–598.

[15] N. Aigerman, Y. Lipman, Injective and bounded distortion mappings in 3D, ACM Trans. Graph. 32 (4) (2013) 106:1–106:14.

[16] G. Cherchi, M. Livesu, R. Scateni, Polycube simplification for coarse layouts of surfaces and volumes, in: Computer Graphics Forum, Vol. 35, 2016, pp. 11–20.

[17] T. K. Dey, J. Sun, Defining and computing curve-skeletons with medial geodesic function, in: Proc. of the Fourth Eurographics Symposium on Geometry Processing, SGP '06, Eurographics Association, Aire-la-Ville, Switzerland, Switzerland, 2006, pp. 143–152.

[18] N. D. Cornea, D. Silver, P. Min, Curve-skeleton properties, applications, and algorithms, IEEE Trans. on Visualization and Computer Graphics 13 (3) (2007) 530–548.

[19] G. Patanè, M. Spagnuolo, B. Falcidieno, A minimal contouring approach to the computation of the reeb graph, IEEE Trans. on Visualization and Computer Graphics 15 (4) (2009) 583–595.

[20] A. Tagliasacchi, T. Delame, M. Spagnuolo, N. Amenta, A. Telea, 3D skeletons: A state-of-the-art report, Computer Graphics Forum.

[21] S. Barbieri, P. Meloni, F. Usai, L. D. Spano, R. Scateni, An interactive editor for curve-skeletons: Skeletonlab, Computers & Graphics 60 (2016) 23–33.

[22] Y. Wang, J. Zheng, Edge based parameterization for tubular meshes, in: ACM International Conference on Virtual-Reality Continuum and Its Applications in Industry, ACM, 2008, p. 23.

[23] T. Huysmans, J. Sijbers, B. Verdonk, Parameterization of tubular surfaces on the cylinder, Journal of WSCG.

[24] M. Zöckler, D. Stalling, H.-C. Hege, Fast and intuitive generation of geometric shape transitions, The Visual Computer 16 (5) (2000) 241–253.

[25] T. Huysmans, J. Sijbers, F. Vanpoucke, B. Verdonk, Improved shape modeling of tubular objects using cylindrical parameterization, in: Medical Imaging and Augmented Reality, Springer, 2006, pp. 84–91.

[26] F. Kälberer, M. Nieser, K. Polthier, Stripe parameterization of tubular surfaces, in: Topological Methods in Data Analysis and Visualization, Springer, 2011, pp. 13–26.

[27] J. Gregson, A. Sheffer, E. Zhang, All-Hex Mesh Generation via Volumetric PolyCube Deformation, Computer Graphics Forum.

[28] K. Hu, Y. J. Zhang, Centroidal voronoi tessellation based polycube construction for adaptive all-hexahedral mesh generation, Computer Methods in Applied Mechanics and Engineering 305 (2016) 405–421.

[29] N. Kowalski, F. Ledoux, P. Frey, Block-structured hexahedral meshes for cad models using 3d frame fields, Procedia Engineering 82 (2014) 59–71.

[30] Y. Zhou, K. Yin, H. Huang, H. Zhang, M. Gong, D. Cohen-Or, Generalized cylinder decomposition, ACM Trans. Graph. 34 (6) (2015) 171:1–171:14.

[31] M. Mortara, G. Patané, M. Spagnuolo, B. Falcidieno, J. Rossignac, Plumber: a method for a multi-scale decomposition

of 3D shapes into tubular primitives and bodies, in: Symposium on Solid modeling and applications, ACM, 2004, pp. 339–344.
[32] J. Wu, L. Liu, Generating quad mesh of 3d articulated shape for sculpting modeling, Journal of Advanced Mechanical Design, Systems, and Manufacturing 6 (3) (2012) 354–365.
[33] L. Antiga, D. A. Steinman, Robust and objective decomposition and mapping of bifurcating vessels, Medical Imaging, IEEE Trans. on 23 (6) (2004) 704–713.
[34] J.-M. Thiery, B. Buchholz, J. Tierny, T. Boubekeur, Analytic curve skeletons for 3D surface modeling and processing, Computer Graphics Forum 31 (7) (2012) 2223–2232.
[35] F. Usai, M. Livesu, E. Puppo, M. Tarini, R. Scateni, Extraction of the quad layout of a triangle mesh guided by its curve skeleton, ACM Trans. Graph. 35 (1) (2015) 6:1–6:13.
[36] C.-Y. Yao, H.-K. Chu, T. Ju, T.-Y. Lee, Compatible quadrangulation by sketching, Computer Animation and Virtual Worlds 20 (2-3) (2009) 101–109.
[37] M. Livesu, A. Muntoni, E. Puppo, R. Scateni, Skeleton-driven adaptive hexahedral meshing of tubular shapes, Computer Graphics Forum 35 (7) (2016) 237–246.
[38] H. Lin, H. Liao, C. Deng, Filling triangular mesh model with all-hex mesh by volume subdivision fitting, Tech. rep., Technical Report, State Key Lab. Of CAD&CG (2012).
[39] Y. Zhang, Y. Bazilevs, S. Goswami, C. L. Bajaj, T. J. Hughes, Patient-specific vascular nurbs modeling for isogeometric analysis of blood flow, Computer methods in applied mechanics and engineering 196 (29) (2007) 2943–2959.
[40] G. B. Arfken, H. J. Weber, F. E. Harris, Mathematical methods for physicists: a comprehensive guide, Acad. Press, 2011.
[41] M. Livesu, F. Guggeri, R. Scateni, Reconstructing the Curve-Skeletons of 3D Shapes Using the Visual Hull, IEEE Trans. on Visualization and Computer Graphics 18 (11) (2012) 1891–1901.
[42] H. Si, Tetgen, a delaunay-based quality tetrahedral mesh generator, ACM Trans. Math. Softw. 41 (2) (2015) 11:1–11:36.
[43] A. Vaxman, M. Campen, O. Diamanti, D. Panozzo, D. Bommes, K. Hildebrandt, M. Ben-Chen, Directional field synthesis, design, and processing, Computer Graphics Forum 35 (2) (2016) 545–572.
[44] L. Lu, B. Lévy, W. Wang, Centroidal voronoi tessellation of line segments and graphs, in: Computer Graphics Forum, Vol. 31, Wiley Online Library, 2012, pp. 775–784.
[45] V. Kolmogorov, R. Zabin, What energy functions can be minimized via graph cuts?, Pattern Analysis and Machine Intelligence, IEEE 26 (2) (2004) 147–159.
[46] Y. Boykov, V. Kolmogorov, An experimental comparison of min-cut/max-flow algorithms for energy minimization in vision, Pattern Analysis and Machine Intelligence, IEEE 26 (9) (2004) 1124–1137.
[47] Y. Boykov, O. Veksler, R. Zabih, Fast approximate energy minimization via graph cuts, Pattern Analysis and Machine Intelligence, IEEE 23 (11) (2001) 1222–1239.
[48] M. Rabinovich, R. Poranne, D. Panozzo, O. Sorkine-Hornung, Scalable locally injective mappings, ACM Trans. on Graphics 36, to appear.
[49] X.-M. Fu, Y. Liu, Computing inversion-free mappings by simplex assembly, ACM Trans. on Graphics (SIGGRAPH Asia) 35 (6).
[50] M. Livesu, R. Scateni, Extracting Curve-skeletons from Digital Shapes Using Occluding Contours, The Visual Computer 29 (9) (2013) 907–916.
[51] M. Livesu, R. Scateni, Practical medial axis filtering for occlusion-aware contours, in: Smart Tools and Apps for Graphics, 2015, pp. 149–154.
[52] A. Tagliasacchi, I. Alhashim, M. Olson, H. Zhang, Mean Curvature Skeletons, Computer Graphics Forum 31 (5) (2012) 1735–1744.
[53] G. Guennebaud, B. Jacob, et al., Eigen v3, http://eigen.tuxfamily.org (2010).
[54] L. Sacht, E. Vouga, A. Jacobson, Nested cages, ACM Trans. Graph. 34 (6).