Neuroinform (2017) 15:247–269
DOI 10.1007/s12021-017-9329-x

ORIGINAL ARTICLE

Generating Neuron Geometries for Detailed Three-Dimensional Simulations Using AnaMorph

Konstantin Mörschel[1] · **Markus Breit**[1] · **Gillian Queisser**[2]

Published online: 26 April 2017

Abstract Generating realistic and complex computational domains for numerical simulations is often a challenging task. In neuroscientific research, more and more one-dimensional morphology data is becoming publicly available through databases. This data, however, only contains point and diameter information not suitable for detailed three-dimensional simulations. In this paper, we present a novel framework, AnaMorph, that automatically generates water-tight surface meshes from one-dimensional point-diameter files. These surface triangulations can be used to simulate the electrical and biochemical behavior of the underlying cell. In addition to morphology generation, AnaMorph also performs quality control of the semi-automatically reconstructed cells coming from anatomical reconstructions. This toolset allows an extension from the classical dimension-reduced modeling and simulation of cellular processes to a full three-dimensional and morphology-including method, leading to novel structure-function interplay studies in the medical field. The developed numerical methods can further be employed in other areas where complex geometries are an essential component of numerical simulations.

✉ Gillian Queisser
gillian.queisser@temple.edu

[1] Goethe Center for Scientific Computing, Goethe University Frankfurt, Kettenhofweg 139, 60325 Frankfurt am Main, Germany

[2] Department of Mathematics, Temple University, 1805 N Broad Street, Philadelphia, PA 19122-6094, USA

Keywords AnaMorph · 3D simulation · NeuroMorpho.Org · Neuron reconstruction · Geometric modeling

Introduction

Numerical simulation has developed into an indispensable tool in modern neuroscience. Various models are now being used to compute the electrical behavior of neurons. While most of these models reduce the spatial dimension to 0 or 1, thus neglecting the physical realm of electrical and biochemical signaling, there is an increasing demand for computational approaches that simulate cell behavior at a highly realistic level. In more recent times, reduced compartmental models have been extended to full three-dimensional models, derived from physical first principles, which are required to investigate the interplay between cellular/intracellular morphology and the underlying biological function. Intra-cellular dynamics are being extended from pure 1D-diffusion/reaction models to 3D models of biochemical signals, that can include complex ion exchange mechanisms across membranes, all the way to electro-diffusion models, represented by the Poisson-Nernst-Planck equations.

At the heart of such full-resolution or hybrid-dimensional computational models lies the computational domain, i.e. the cellular and intra-cellular architecture. Systematic anatomical reconstruction of neuron morphology has brought forth large databases, such as NeuroMorpho.Org, where semi-manual reconstruction data of neuron morphologies are stored in point-diameter files. While these graph-based data are sufficient for compartmental models (one-dimensional in space), they lack the surface and volume

representation needed in simulating three-dimensional biophysical models.

In this paper, we therefore developed a framework for generating three-dimensional surface representations from existing anatomical, one-dimensional, cellular reconstructions. One major asset of this approach is that existing reconstruction data can be recycled to compute an equivalent three-dimensional representation, which can be used in either full three-dimensional computations or even in hybrid 1D/3D approaches, where the electrical signal is rapidly computed in 1D and mapped to a 3D model of coupled biochemical events. The AnaMorph framework is based on non-linear piecewise analytical modeling, using the existing reconstruction data as the "spine" for the computed canal-surfaces of dendritic and axonal arbors of neurons. Aside from computing consistent surface meshes from these analytical representations, AnaMorph can further be used as a quality control framework for semi-manual anatomical reconstructions. After a brief motivation example, we will introduce the general geometric, analytic and algorithmic concepts of AnaMorph and we will show results that can be used within advanced numerical simulations.

We will give a brief example of a hybrid-dimensional numerical simulation of intra-cellular calcium dynamics, motivating the development of methods and algorithms presented in this paper. Using AnaMorph, we generate a three-dimensional computational grid from a retina ganglion cell of the mouse (NMO_05371, Coombs et al. 2006), retrieved from the NeuroMorpho.Org database (Ascoli 2006). The model consists of a simple diffusion-reaction equation for intra-cellular calcium u and a buffer b in the three-dimensional domain Ω:

$$\frac{\partial u}{\partial t} = \operatorname{div}(D_u \nabla u) - k^+ u \cdot b + k^-(b^{\text{tot}} - b) \quad \text{on } \Omega \tag{1}$$

$$\frac{\partial b}{\partial t} = \operatorname{div}(D_b \nabla b) - k^+ u \cdot b + k^-(b^{\text{tot}} - b) \quad \text{on } \Omega \tag{2}$$

D_u and D_b define the diffusion coefficients of calcium and the buffer, respectively. Values $k^{\pm}$ are binding/unbinding constants of the reaction of calcium and the buffer b, while b^{tot} denotes the total buffer concentration, i.e., the sum of bound and unbound form, which is supposed to be constant. The boundary conditions for these equations are defined by channels embedded in the plasma membrane $\partial\Omega$. These are calcium concentration and membrane potential (V_m)-dependent and are computed as nonlinear Neumann calcium fluxes across the membrane:

$$\frac{\partial u}{\partial \mathbf{n}} = \mathbf{j}(V_m(\mathbf{x}, t), u(\mathbf{x}, t), \mathbf{x}, t) \quad \text{on } \partial\Omega \tag{3}$$

where j is computed by voltage-gated calcium channel models found in, e.g. Borg-Graham (1999). The computation of V_m is performed on the associated one-dimensional computational domain ω. This domain is compartmentalized and on every compartment, the capacitor equation

$$C_m \frac{\partial V_m}{\partial t} = I_m + I_{ax} \tag{4}$$

is imposed, where C_m is the compartment capacitance, I_{ax} defines the axial currents and I_m the (inward) membrane current as the sum of all relevant ionic currents including, in our case, sodium, potassium and calcium ions. In the numerical simulation, V_m is calculated using the method described in Breit et al. (2016) and then mapped from ω onto $\partial\Omega$ as described in Grein et al. (2014), where it is used to compute the calcium membrane fluxes.

Sample simulation results are shown in Fig. 1. Note that in order to couple 1d and 3d models, the construction of a numerically feasible and equivalent domain Ω to a given 1d neuron representation ω is essential and has guided the development of AnaMorph. The significance of the three-dimensional intracellular architecture has been demonstrated, e.g., in Wittmann et al. (2009) and Queisser et al. (2011). In a previous paper, one of the authors studied and validated coupled 1D/3D simulation methods (Grein et al. 2014), in which a Hodgkin-Huxley derived electrical model in 1D was computed with NEURON (Hines and Carnevale 1997) and coupled to a 3D biochemical model for calcium, computed with UG 4 (Vogel et al. 2013).

The AnaMorph Framework

The input for AnaMorph are anatomically reconstructed morphology data sets such as the standardized SWC files publicly accessible in large quantities in the growing database NeuroMorpho.Org.

Such reconstructions can be interpreted as a directed graph $\mathcal{T} = (V, E)$, called *morphology tree*, whose vertices $(t, p, r) \in V$ are points $p \in \mathbb{R}^3$ along with a radius $r \in \mathbb{R}^+$ and a type t identifying the cellular component (soma: $t = 1$, axon: $t = 2$, basal or apical dendrite: $t = 3, 4$). The data semantics depend on the type t: for axon and dendrite (neurites), inspected sections are recorded by an approximate center point p and an average radius r, whereas soma vertices can be interpreted in various ways (see "Discussion"). The topology of $\mathcal{T}$ (given by E, where in $e = (u, v) \in V^2$ the vertex u denotes the parent of v) represents the cellular branching structure. Vertices $v \in V$ are clustered into soma, axon and dendrite vertices, where the term neurite vertex is

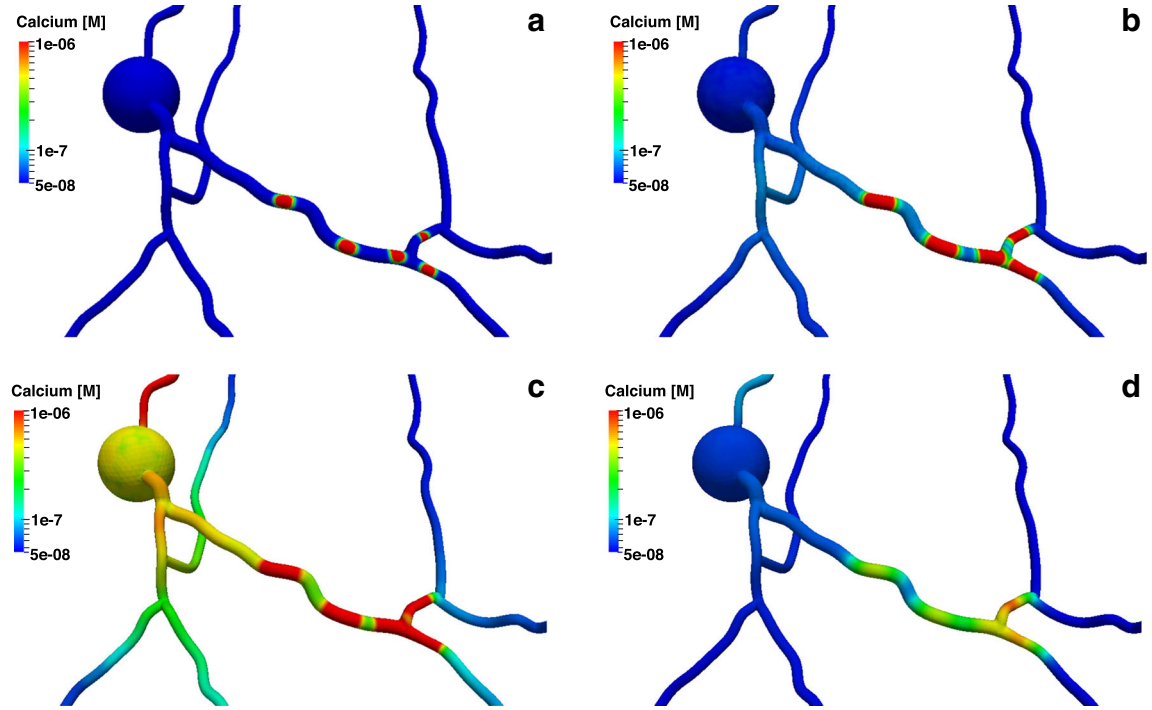

Fig. 1 Example of numerical simulation of intra-cellular calcium dynamics coupled to the membrane potential equation using the frameworks presented in Breit et al. (2016) and Grein et al. (2014). The three-dimensional morphology and surface triangulation was carried out using AnaMorph. Three-dimensional simulations were performed with UG 4 (Vogel et al. 2013). A stimulation pattern was applied in the form of spatio-temporal Neumann boundary conditions. The subfigures show the intracellular calcium concentration 0.1 mm (**a**), 1.4 mm (**b**), 3.7 mm (**c**) and 40.0 mm (**d**) after the onset of stimulation

used to represent both axon and dendrite vertices. $v \in V$ is called *simple* if outdeg $(v) = 1$, *branching* if outdeg $(v) > 1$ and *terminal* of outdeg $(v) = 0$. A neurite vertex whose parent is the soma vertex s is called *neurite root vertex*. An edge $e = (u = (t, p_u, r_u), v = (t, p_v, r_v)) \in E$ connecting two neurite vertices is called *neurite segment*, its *length* is given by $l(e) = \|p_v - p_u\|$ and its *angle* shall be defined as $\angle(e) = \arccos((p_u - p_f) \cdot (p_v - p_u))$ where $f = (t_f, p_f, r_f)$ is the parent vertex of u in $\mathcal{T}$. The angle of e satisfies $0° \leq \angle(e) \leq 180°$, where $\angle(e) = 0$ iff p_f, p_u and p_v lie on a straight line. The *radius ratio* of e is defined as $R_r(e) = \max\{r_u/r_v, r_v/r_u\}$ with $R_r(e) \geq 1$. Although embedded in $\mathbb{R}^3$, morphology trees are inherently one-dimensional structures and do not provide direct access to a three-dimensional surface representation. AnaMorph implements a chain of algorithms to convert $\mathcal{T}$ into a biologically plausible high quality polygonal surface mesh representation of the plasma membrane exportable to popular surface formats such as wavefront .obj, which can be imported and inspected by software such as ProMesh (Reiter 2012, 2014) or Meshlab (Cignoni et al. 2008). The processing chain consists of the steps (cf. flow chart in Fig. 2).

(1) preconditioning the morphology tree $\mathcal{T}$ constructed from input file ("Morphology Preconditioning"),
(2) generation of a non-linear piecewise defined geometric model from $\mathcal{T}$ ("Geometric Modeling"),
(3) computational analysis of the geometric model using analytically derived geometric intersection criteria that are verifiable with robust numerical solvers ("Consistency Analysis") and
(4) algorithmic generation and optimization of triangular polygonal surface meshes from said model, provided the analysis has confirmed the model to be geometrically valid ("Mesh Generation").

Special focus has been placed on deriving formal, reliable guarantees for the topological and geometric integrity of the generated meshes: the output meshes are guaranteed to be orientable, topological 2-manifolds without boundary of genus zero and geometrically free of self-intersections. Both these requirements are critical for their use in advanced numerical simulation. After the initial meshing, a two-stage mesh post-processing chain is applied, which produces final output meshes of consistently high triangle quality in terms

of aspect ratio (Fig. 3 shows part of a mesh comprising multiple branching neurites).

Morphology Preconditioning

This step needs to be understood in the light of the geometric modeling described in "Geometric Modeling". To keep it brief here and to leave the details to be found in the modeling section, let it be said that the geometric model will consist of interpolating the vertices given in the input file using a spatial polynomial spline curve. The plasma membrane surface will then be defined by inflating this curve (using the radius information) around the axis given in each point by the direction of the curve.

A hurdle for this interpolation approach is posed by neurite segments whose length is small compared to the radii of their delimiting vertices. They tend to create sharp angles where the inflated spline curve surfaces are likely to be self-intersecting. Additionally, irregularly spaced neurite points, i.e., large variations in neurite segment length, make interpolation difficult. Both very short neurite segments as well as generally irregular point sampling density can be found in many morphologies from NeuroMorpho.Org. We have therefore chosen to formalize the notion of "short" neurite segments using two different violation categories.

Definition 1 (Minimal distance violations)
Let $e = \{u, v\}$ be an edge with vertex coordinates p_u, p_v; radii r_u, r_v and length $l(e) = \|p_v - p_u\|$.

(a) For $\alpha \geq 2$, e is said to exhibit an *α-primary minimal distance violation* (α-PMDV), if $l(e) \leq \alpha \max\{r_u, r_v\}$.
(b) For the definition of a second type of short neurite segment e, consider the neighbor sets $U_{\bar{v}} = \{u\} \cup N(u) \setminus \{v\}$ and $V_{\bar{u}} = \{v\} \cup N(v) \setminus \{u\}$, where $N(u)$ denotes the set of neighbors of u in the undirected version of $\mathcal{T}$ (note that $u \notin V_{\bar{u}}$ and $v \notin U_{\bar{v}}$). Additionally, let $R_U = \max\{r_x \mid (t_e, p_x, r_x) \in U_{\bar{v}}\}$, if u is not a neurite root vertex, and $R_U = 0$ otherwise. Similarly, let $R_V = \max\{r_x \mid (t_e, p_x, r_x) \in V_{\bar{u}}\}$ if v is not a terminal vertex, and $R_V = 0$ otherwise. Then for $\beta \geq 1$, e is said to exhibit a *β-secondary minimal distance violation* (β-SMDV), if $l(e) \leq \beta(R_U + R_V)$.

Aside from reducing the likelihood of self-intersecting model surfaces, eliminating all minimal distance violations from the morphology tree will help rule out false positives in the intersection analysis (cf. "Intersection Analysis").

Since minimal distance violations (MDV) are present in virtually all morphologies, the issue has been specifically addressed using a morphology preconditioning algorithm: Algorithm 1 aims at eliminating MDVs and regularizing the morphology data before applying the geometric model.

Algorithm 1 Greedy preconditioning for morphology trees

Input: $\mathcal{T} = (V, E)$: Morphology tree, $\alpha \geq 2$, $\beta \geq 1$: MDV parameters, $\gamma \geq 2$: threshold parameter.
Output: $\mathcal{T}' = (V', E')$: Preconditioned morphology tree.In this algorithm, e shall always refer to a neurite segment $e = (u = (t, p_u, r_u), v = (t, p_v, r_v)) \in E$.

(1) Initialize an empty min-heap (priority queue) Q. For every $e \in E$, where $r^+ = \max\{r_u, r_v\}$ and R_U, R_V are defined as in Def. 1, compute a key value

$$\text{key}(e) = \min\{\|p_v - p_u\| - \alpha r^+, \|p_v - p_u\| - \beta(R_U + R_V)\} \quad (5)$$

and insert e with key (e) into Q.
(2) While Q is not empty, delete the min element $e = (u, v)$ from Q. If e does no longer exist in T or key $(e) \geq 0$, discard e. Otherwise:
  (2a) If both u and v are simple vertices, then e is *collapsed* into a new neurite vertex w into the new vertex $w = (\frac{1}{2}(p_u + p_v), \frac{1}{2}(r_u + r_b), t)$. The topology of $\mathcal{T}$ is updated as follows: If f_u and c_v denote the parent of u and the only child of v, respectively, then the three neurite segments containing either u or v are removed and replaced by two neurite segments $n_0 = (f_u, w)$ and $n_1 = (w, c_v)$. Subsequently, n_0 and n_1 are inserted into Q with their respective keys. Note that the three deleted neurite segments are still indexed in Q, but can be identified and deleted lazily once they are dequeued.
  (2b) Otherwise, if u is a branching or neurite root vertex and v is a simple vertex, then v is deleted from T. The topology of the morphology network is updated accordingly.
  (2c) Otherwise, if u is a simple vertex and v is a branching or terminal vertex, then u is deleted from T. The topology of the morphology network is updated accordingly.
(3) If $\gamma < \infty$, then for all neurite segments e: if $l(e) > \gamma \max(r_u, r_v)$, split e into a minimum number k of neurite segments of equal length L such that $L < \gamma \max(r_u, r_v)$. The splitting is performed by inserting new neurite vertices on the line segment connecting p_u and p_v, where radii are linearly interpolated on the line segment, and updating the topology of $\mathcal{T}$ in the obvious way.

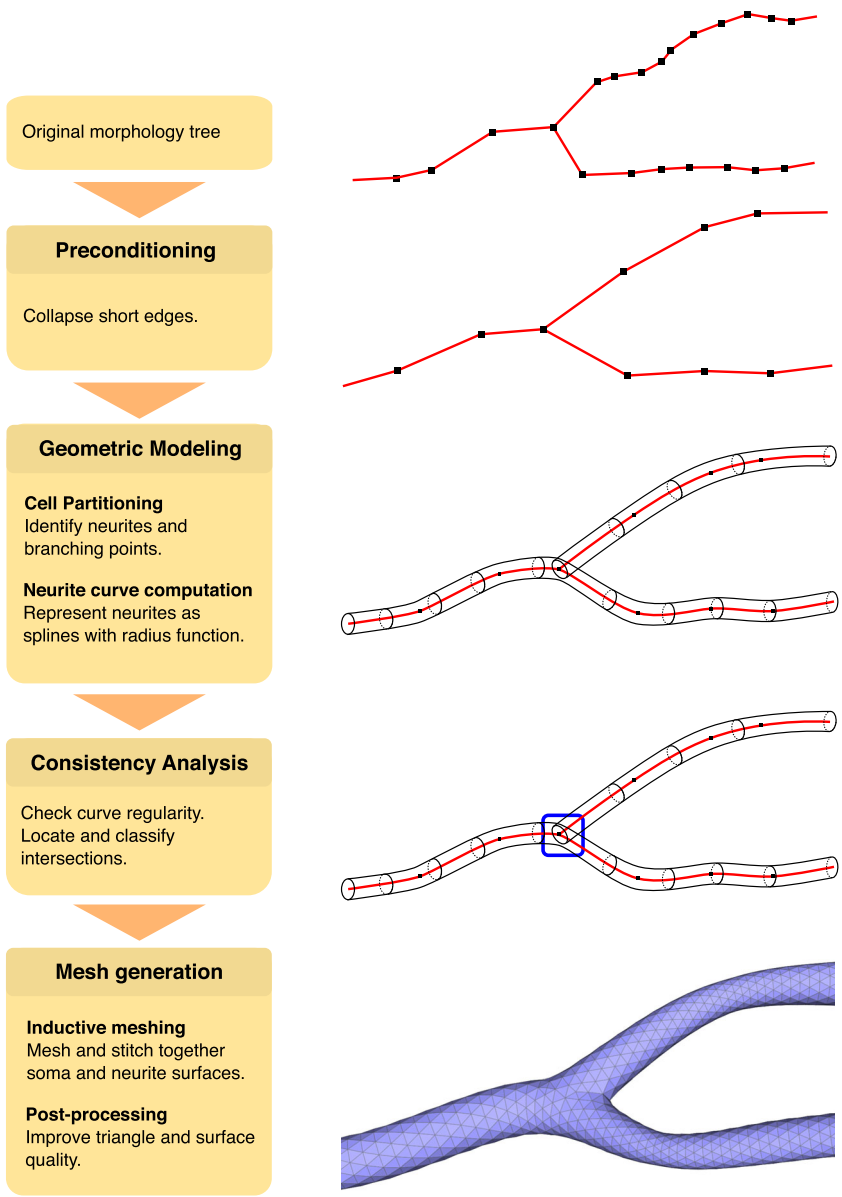

Fig. 2 Flow chart depicting the stages of the AnaMorph processing chain. Starting with the original morphology tree, i.e., anatomical 1D graph reconstructions that contain point and diameter information of the cell, a preconditioning step collapses short edges. On the preconditioned point-diameter data, a geometric modeling step performs cell partitioning to identify single neurites and all branching points in the cell. Each neurite can then be represented by a spline interpolation function with attached radius information. Consistency analysis checks for curve regularity and identifies and classifies intersections, e.g. self-intersections. In a final step, the geometric model is used as a basis for inductive mesh generation to mesh and unify all neurites and the soma. A final post-processing step improves the surface mesh quality for numerical simulation

It was deliberately chosen to delete or collapse only simple neurite vertices. This preserves the branching topology of $\mathcal{T}$, in particular, minimal distance violating neurite segments connecting two closely spaced branching vertices are not modified. If the input morphology network $\mathcal{T} = (V, E)$ does not contain any neurite segments with an α-PMDV or β-SMDV, then Algorithm 1 with the threshold parameter $\gamma = \infty$ does not change $\mathcal{T}$ at all. By letting $\gamma < \infty$, step (3) can be used to split long neurite segments in order to reduce neurite segment length variance. Since neurite segments are collapsed into the average vertex (5), the algorithm bears similarities to diffusion-based smoothing. Additionally, it specifically targets the shortest most critical remaining MDV using the greedy approach. The user can directly influence the level of smoothing by altering the parameters α and β. Though simple in structure, Algorithm 1 already produces very good results. See Fig. 4 for its effect on the geometric model of an extremely densely measured morphology. The applicability of the geometric model is significantly improved, albeit at the cost of data resolution.

Geometric Modeling

Prior to meshing, a morphology tree $\mathcal{T}$ is converted to an analytic representation consisting of multiple individual

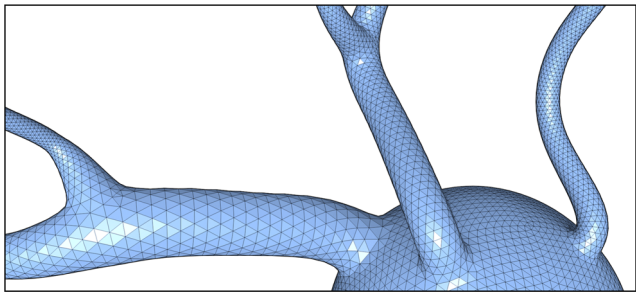

Fig. 3 Close-up of the soma region in a post-processed cell mesh

modeling surfaces. The soma $s = (1, p_s, r_s)$ is modeled as the sphere

$$S_{r_s}(p_s) := \left\{ x \in \mathbb{R}^3 \mid \|x - p_s\| = r_s \right\}. \tag{6}$$

Neurites are modeled by a non-linear interpolation approach instead of the simple piecewise linear cone frusta approach widely used due to its simplicity. Note that the non-linear description does not include any branching points, but is applicable only to each neurite individually. The topological neurite/neurite or neurite/soma connections are only realized during the meshing process. To illustrate the principle, consider a non-branching axon $A = (v_0, \ldots, v_n)$. The model for the plasma membrane surface is constructed using a (continuously differentiable) curve γ going through the vertices $v_0, \ldots, v_n$ in the correct order. This curve is expanded at each position and in all directions perpendicular to the curve by a radius given by some (continuous) function ρ interpolating the given radii. For the simple case where all vertices are located on a straight line, one could choose γ as the straight line and ρ as the linear interpolation between the vertices, resulting in a cone frustum description of the surface.

A more formal description of the general surface is given in the following definition.

Definition 2 (Canal surface)
To given neurite vertices $(v_0 = (s, p_0, r_0), \ldots, v_n = (s, p_n, r_n))$, a *canal surface* $\mathfrak{C}$ is defined by any pair (γ, ρ) with

$$\gamma : T = [t_0, t_n] \to \mathbb{R}^3, \tag{7}$$
$$\rho : T = [t_0, t_n] \to \mathbb{R}^+ \tag{8}$$

fulfilling

(i) γ is of class C^1 and regular, i.e., $\frac{d\gamma}{dt}(s) \neq 0 \ \ \forall s \in T$,
(ii) ρ is of class C^0 and
(iii) γ and ρ are interpolating, i.e., $\exists\, t_0 < t_1 < \cdots < t_{n-1} < t_n : (p_i, r_i) = (\gamma(t_i), \rho(t_i)),\ 0 \leq i \leq n$

as the union of circles of radius ρ around γ in the sense that

$$\mathfrak{C} = \mathfrak{C}(\gamma, \rho) = \bigcup_{t \in T} \left\{ x \in \mathbb{R}^3 : \ (x - \gamma(t)) \perp \dot{\gamma}(t),\ \|x - \gamma(t)\| = \rho(t) \right\}. \tag{9}$$

If $\rho = r \in \mathbb{R}$ is constant, then the surface is called a *pipe surface* and denoted $\mathfrak{P} = \mathfrak{P}(\gamma, r)$.

The AnaMorph modeling algorithm implements a canal surface consisting of cubic spline interpolations for each dimension component of γ – thus even attaining γ to be of class C^2 – and piecewise linear (with respect to the arc length of γ) interpolation for ρ.

From an algorithmic point of view, a parametric definition of the surface as the image of the function

$$\mathfrak{c}(\gamma, \rho) : T \times [0, 2\pi) \ \to \ \mathbb{R}^3,$$
$$(t, \theta) \ \mapsto \ \gamma(t) + \rho(t)\,(\cos(\theta)\mathbf{n}(t) + \sin(\theta)\mathbf{b}(t)) \tag{10}$$

proves to be useful, where $\mathbf{n}(t)$ and $\mathbf{b}(t)$ denote orthogonal unit vectors spanning the curve's normal plane at the location $\gamma(t)$ and θ is an angle parametrizing circles in these planes. Function (10) permits a mesh discretization of $\mathfrak{C}$

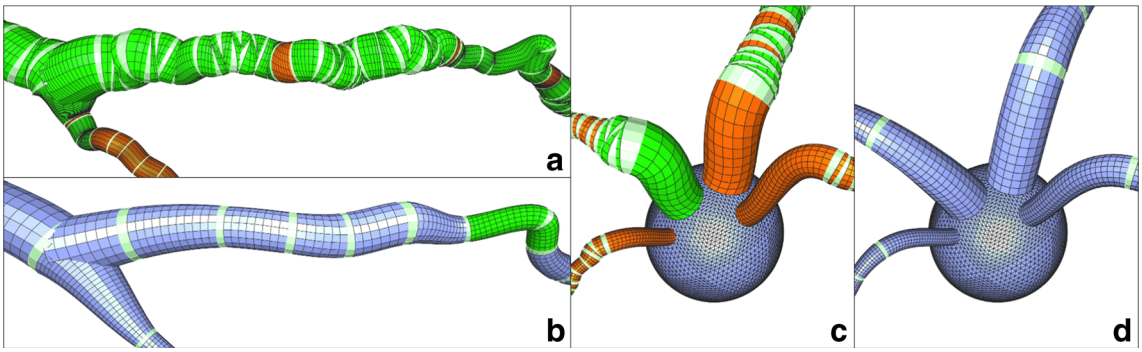

Fig. 4 Effect of the preconditioning Algorithm 1 with $\alpha = 2.0$, $\beta = 1.0$, $\gamma = \infty$ on morphology NMO_04186 (McDonald et al. 2007) from NeuroMorpho.Org featuring dense irregular data. Modeling surfaces for the original data (**a**, **c**) and after preconditioning (**b**, **d**). The colors encode intersection types (cf. "Intersection Classification"), *white*: segment borders, *green/orange*: self-intersections, *blue*: free of intersections. Views rendered by a separate canal surface visualization tool

by carefully sampling the parametric domain $T \times [0, 2\pi)$ to obtain vertices which are then suitably connected with consistently oriented triangular faces.

The rest of this section will deal with the details of the geometric modeling.

Cell Partitioning

The above canal surface modeling can be applied to any path $P = (v_0, \ldots, v_n)$ of consecutive neurite vertices in the input morphology tree $\mathcal{T} = (V, E)$. Such a path P is called *neurite path*, its *chord length* is defined by $L(P) = \sum_{i=1}^{n} \|p_i - p_{i-1}\|$. However, branching of dendrites or axons introduces ambiguity with regard to the question of which branch belongs to which neurite, as only one branch can consecutively extend the original neurite. Thus, it is necessary to provide some specification. To that end, in the first phase of geometric modeling, the morphology tree $\mathcal{T}$ is partitioned into a set of neurite paths such that every neurite segment $e \in E$ is part of exactly one path.

Every neurite path is constructed in such a way that it starts in a neurite root vertex or in a branching vertex and ends in a branching vertex or in a terminal vertex. The partitioning process can be formulated as modified depth-first traversals of $\mathcal{T}$ starting at the root vertices. At each branching vertex, it must be decided whether the current path is to end there and, if not, which of the children will continue the path. This choice matters for the local shape of the branching point, due to the regularity imposed by the canal surface.

In AnaMorph, the generic structure of the algorithm is as follows:
Let b be the currently visited branching vertex and let C be the set of its children.

(1) User-definable thresholds $\angle_{max}$ and R_{max} control whether a child $c \in C$ of b is filtered out, i.e., rejected as successor candidate. This is the case if

$$\angle(b, c) \geq \angle_{max} \quad \vee \quad R_r(b, c) \geq R_{max},$$

i.e., if c branches from b at too big an angle or if the radius changes too much from b to c. If all children are filtered out then the current path ends at b.

(2) Let $C' \neq \varnothing$ be the set of children that are not filtered out. The following partitioning strategies are currently available in AnaMorph:

(a) Maximum chordal depth (the default strategy): Return $w \in C'$ such that the chord length of the path from w to a terminal vertex in the sub-tree rooted in w is maximized.
(b) Greedy minimal angle: Return $w \in C'$ such that $\angle(v, w)$ is minimized.
(c) Greedy maximal radius: Return $w \in C'$ of maximal radius.
(d) A class of penalty-based functions with user-definable parameters. The successor candidate with minimal penalty is selected.

The chordal depth and minimum angle strategies have been found to generally produce good results in practice.

Neurite Curve Computation

Once the partitioning has been computed, each obtained neurite path is modeled individually as a piecewise defined canal surface $\mathfrak{C}_P$ as described in the following.
Let $P = \{v_0 = (t_P, p_0, r_0), \ldots, v_n = (t_P, p_n, r_n)\}$ be a neurite path and let $e_i = (v_i, v_{i+1})$ denote its i-th neurite segment. Firstly, a curve γ_P for the entire path P is computed (we call γ_P *neurite curve*), using interpolating, piecewise defined parametric curves with polynomial component functions. Other options, such as approximating curves, are available within the presented theoretical and algorithmic framework, the only conditions being that

(i) γ_P is of class C^1 at least and regular,
(ii) γ_P has (piecewise) polynomial component functions,
(iii) γ_P contains all branching points and neurite root points on the path P.

Spline data interpolation research has brought forth various notions of measuring the "fairness" or quality of an interpolating or approximating spline, in part derived from physical analogies like minimum energy functionals, see, e.g., Greiner (1991) or Levien and Séquin (2009). A common choice for one-dimensional data interpolation are cubic splines, where Hermite boundary conditions are relevant in the discussed context.

Many morphologies from NeuroMorpho.Org produce very long neurite paths (much more than 100 segments). Furthermore, many morphologies are problematic in that they encode biologically infeasible geometries. In order to rectify such data, be it semi-manually or algorithmically, it becomes necessary to modify and frequently re-evaluate the geometric model, including neurite curve computation. To avoid intolerably long run-times, cubic splines are chosen as default in AnaMorph, because they offer an attractive compromise between computational complexity and result quality. As long as the above conditions for γ_P hold, the modular implementation design allows alternative methods to be added with minimal effort.

The theory of one-dimensional cubic splines readily translates to parametric (space) curves. For now, we assume that there is already a parametrization for P, i.e., a grid of parameter values $\Delta_P : a = t_0 < \ldots < t_n = b$ for the points p_i, as well as two tangent boundary condition

vectors $\gamma_0' = (\gamma_{0,x}', \gamma_{0,y}', \gamma_{0,z}')^T$ and $\gamma_n' = (\gamma_{n,x}', \gamma_{n,y}', \gamma_{n,z}')^T$ (how to obtain these will be discussed in a moment). By writing the i-th neurite point as $p_i = (x_i, y_i, z_i)^T$, the one-dimensional cubic spline computation can be generalized to parametric curves simply by choosing the three component functions S_x, S_y and S_y of $\gamma_P \colon [a, b] \to \mathbb{R}^3,\ t \mapsto (S_x(t), S_y(t), S_z(t))^T$ to be cubic splines with respect to the common parameter grid Δ_P as follows: $S_x(t)$ is chosen as the cubic spline for the data values $(t_i, x_i),\ 0 \leq i \leq n$ with Hermite boundary conditions $S_x'(t_0) = \gamma_{0,x}'$ and $S_x'(t_n) = \gamma_{n,x}'$. $S_y(t)$ and $S_z(t)$ are chosen accordingly. The obtained curve γ_P is of order C^2 on $[a, b]$ and furthermore, a characteristic curvature-related optimality property of the component functions carries over to the curve γ_P in that it minimizes the functional

$$I_c(\xi) = \int_a^b \left\| \ddot{\xi}(t) \right\|^2 dt \tag{11}$$

among all C^2 curves interpolating $p_0, \ldots, p_n$ and respecting the imposed Hermite boundary conditions. If a curve $\gamma(s)$ is parametrized by arc length, its curvature is given by $\kappa(s) = \left\| \gamma''(s) \right\|$, whereas $\kappa(t) = \|\dot{\gamma}(t) \times \ddot{\gamma}(t)\| \|\dot{\gamma}(t)\|^{-3}$ holds for an arbitrary parametrization $\gamma(t),\ t \in [a, b]$. Now if the parametric speed $\|\dot{\gamma}(t)\|$ is close to one, then $\|\ddot{\gamma}(t)\|$ is a good approximation of the curvature of γ and thus the functional (11) can be considered an approximate measure for the total curvature of the curve ξ in $[a, b]$. The problem of finding a suitable parametrization grid Δ_P and the two boundary tangent vectors γ_0' and γ_n' from the given neurite points $p_i \in \mathbb{R}^3,\ 0 \leq i \leq n$ is an active area of research, see Floater and Surazhsky (2006). The parametric domain is chosen as $[a, b] = [0, 1]$ to simplify subsequent numerical computations. Three parametrization strategies have been implemented:

(1) Uniform parametrization: $t_i = \frac{i}{n},\ 0 \leq i \leq n$.
(2) Chordal parametrization: $t_0 = 0$, $t_i = \frac{L_i}{L(P)}$, where $L_i = \sum_{k=1}^{i} \|p_k - p_{k-1}\|,\ 1 \leq i \leq n$, and $L(P)$ is the chord length of P.
(3) Centripetal parametrization: $t_0 = 0$, $t_i = \frac{C_i}{C_P}$, where $C_i = \sum_{k=1}^{i} \sqrt{\|p_k - p_{k-1}\|},\ 1 \leq i \leq n$, and $C_P = \sum_{k=1}^{n} \sqrt{\|p_k - p_{k-1}\|}$.

In case of uniform parametrization, the values $t_{i+1} - t_i$ are constant and thus the parametric speed depends on the distances $d_i = \|p_{i+1} - p_i\|$. If these vary greatly, high parametric acceleration may lead to "overshooting" of the neurite curve, e.g., in case of a long neurite segment followed by a sharp turn (cf. Fig. 5b). Chordal and centripetal parametrization both take the distances d_i into account and thus have the tendency to produce smaller variations in parametric speed. However, in case of a sharp turn, it would usually be ideal to slow down, and so both approaches have their drawbacks. The chordal parametrization is (up to the constant factor $L(P)$) the arc length parametrization on the polyline obtained by connecting all consecutive points p_i, p_{i+1} of P with straight line segments and thus it tends to effect a more or less constant parametric speed of γ_P. It can only be speculated what the original neuron looked like and it is therefore hard to quantify which method is the "best" for the modeling purpose, but the chordal parametrization has produced very good results during testing and has been chosen as the default strategy. A comparison of all three strategies in an engineered test case is compiled in Fig. 5.

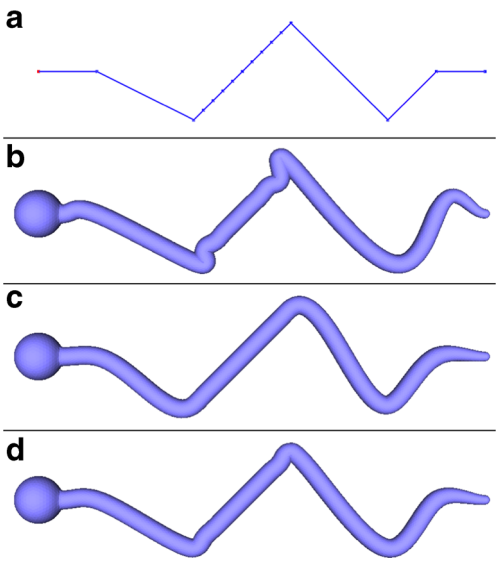

Fig. 5 Impact of the spline parametrization strategy. A suitable test geometry has been constructed in SWC format (**a**). AnaMorph's output (without preconditioning) with uniform (**b**), chordal (**c**) and centripetal (**d**) parametrization strategy. Note the severe overshooting at the transition from coarse to fine resolution in the uniform case. The chordal strategy also leads to overshooting, but in the other direction, which does not cause correctional bending as in the former case

The boundary tangent vectors γ_0' and γ_n' are generally chosen as $\gamma_0' = \lambda_0(p_1 - p_0)\|p_1 - p_0\|^{-1}$ and $\gamma_n' = \lambda_n(p_n - p_{n-1})\|p_n - p_{n-1}\|^{-1}$ with two positive scaling factors $\lambda_0, \lambda_n \in \mathbb{R}^+$. There is one exception: If P is a neurite root path, then p_0 is a neurite root point that lies on the surface of a soma sphere $S_{r_s}(c)$ with center $c \in \mathbb{R}^3$. Then γ_0' is chosen as the sphere normal $\gamma_0' = \lambda_0(p_0 - c)(\|p_0 - c\|)^{-1}$ to enforce that neurites align perpendicular to the soma.

With parametrization and boundary conditions chosen, the piecewise defined cubic spline curve γ_P can be computed in linear time and space by solving three tridiagonal systems.

Together with the radius function ρ_P for P, the neurite path P as a whole is then represented by the piecewise defined canal surface $\mathfrak{C}(\gamma_P, \rho_P)$ in the geometric model (see Def. 2).

The radius function $\rho_P(t)$ is defined by linear interpolation of the radii $r_0, \ldots, r_n$ with respect to the arc length of γ_P.

Geometric modeling is complete after the soma and all neurite paths of the partitioning have been processed. The set of modeling surfaces can then be analyzed for intersections in the consistency analysis and linked to a manifold during mesh generation.

Consistency Analysis

AnaMorph has been designed to give reliable guarantees for the topological and geometric integrity and the biological plausibility of the generated output meshes. This requires a detailed analysis of the geometric model prior to meshing. If a morphology tree encodes biologically infeasible or problematic geometry, the respective parts of the input data (often only a very small fraction) are identified precisely. This significantly simplifies a potential data rectification process compared to semi-manual re-inspection of the entire data set. If, on the other hand, the morphology is found to be "clean", the desired formal guarantee can be given.

In "Intersection Analysis", we will first discuss the intersection analysis phase, where several types of geometric intersections on and in-between the modeling surfaces are identified using analytical problem formulations that allow a robust numerical solution. The employed analytical derivations are in part based on Patrikalakis and Maekawa (2002) and Maekawa et al. (1998). The results obtained during intersection analysis are then further processed and interpreted during the intersection classification phase described in "Intersection Classification", key issue being the derivation of a set of conditions to distinguish biologically feasible from infeasible intersections.

Intersection Analysis

All geometric intersection problems considered during analysis can be reduced to two types of algebraic problems:

(1) Finding all roots of a univariate polynomial on a closed interval $I = [x_0, x_1] \subset \mathbb{R}$, i.e., solving

$$p(x) = 0, \quad x \in [x_0, x_1]. \tag{12}$$

(2) Finding all solutions to a homogeneous system of two bivariate polynomials $p(x, y), q(x, y)$ on a closed rectangular domain $A = [x_0, x_1] \times [y_0, y_1] \subset \mathbb{R}^2$, i.e., solving

$$p(x, y) = q(x, y) = 0, \quad (x, y) \in [x_0, x_1] \times [y_0, y_1]. \tag{13}$$

For further discussion, it is required and will be ensured that $p - q$ does not possess a non-trivial polynomial divisor, i.e., the solution set of Eq. 13 is required to be finite. As the arising root-finding problems do not admit an analytic solution, AnaMorph implements adapted, robust and globally convergent numerical solvers based on the Bernstein-Bézier representation of univariate and bivariate polynomials. For the univariate case, we use the Bézier-Clipping algorithm (Sederberg and Nishita 1990; Schulz 2009) and the Quadratic-Clipping algorithm (Bartoň and Jüttler 2007a), for the bivariate case, we use Bivariate Linear-Clipping (Bartoň and Jüttler 2007b) and Bivariate Quadratic-Clipping (Jüttler and Moore 2011). We refer to Farouki and Rajan (1988), Farouki (2012), Jüttler (1998), Liu et al. (2009), and Farouki and Goodman (1996) for more information.

For the rest of this section, let $c = (u = (t_c, p_u, r_u), v = (t_c, p_v, r_v))$ and $d = (w = (t_d, p_w, r_w), z = (t_d, p_z, r_z))$ be two neurite segments from the analysed morphology tree $\mathcal{T}$, $\gamma\colon [x_0, x_1] \to \mathbb{R}^3$ and $\delta : [y_0, y_1] \to \mathbb{R}^3$ the corresponding curve segments from the geometric model and $\mathfrak{C}_c = \mathfrak{C}(\gamma, \rho_c)$ and $\mathfrak{C}_d = \mathfrak{C}(\delta, \rho_d)$ the respective segments of the neurite canal surfaces.

Computationally, it is rather difficult to determine whether general canal surface segments intersect. This is why AnaMorph does not check for intersections on the canal surface segments directly, but on capsule-shaped bounding volumes for which the computation is easy. More specifically, the canal surface is first approximated by the bounding pipe surface $\mathfrak{P}_c = \mathfrak{P}(\gamma, r_c^+ = \max\{r_u, r_v\})$ for c. Subsequently, the volume $V(\mathfrak{P}_c)$ of this pipe surface $\mathfrak{P}_c$ is defined as

$$V(\mathfrak{P}_c) = \bigcup_{x \in [x_0, x_1]} B_{r_c^+}(\gamma(x)). \tag{14}$$

where $B_{r_c^+}(\gamma(x)) \subset \mathbb{R}^3$ is the closed ball of radius r_c^+ and center $\gamma(x)$ (cf. schematic in Fig. 6). Since radii are interpolated between r_u and r_v in a monotone fashion for the canal surface, $\mathfrak{P}_c$ fully encloses $\mathfrak{C}_c$. For most morphologies from NeuroMorpho.Org, radii do not exhibit large jumps from one vertex to the next and so the bounding volume approximation is generally tight. Using bounding capsules instead of the original canal surface segments during intersection analysis will therefore be relatively unlikely to report false positives while at the same time considerably simplifying

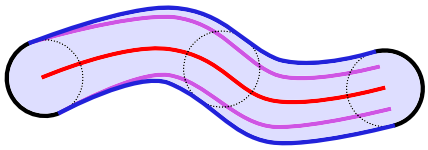

Fig. 6 Two-dimensional schematic of the volume $V(\mathfrak{P}_c)$ of a pipe surface segment $\mathfrak{P}_c$: a segment of the neurite curve (*red*), the associated canal surface segment (*magenta*), the bounding pipe surface (*dark blue*) and the bounding capsule volume (*light blue*)

the task. This allows for a very efficient analysis of even the largest morphologies from NeuroMorpho.Org.

For the rest of this section, let $\Gamma = \mathfrak{P}_c$ and $\Delta = \mathfrak{P}_d = \mathfrak{P}(\delta, r_d^+)$ be two distinct pipe surfaces.

Neurite-Neurite Intersections In order to check for intersections of two distinct pipe surfaces Γ and Δ (Fig. 7a), we check whether their volumes (as defined by Eq. 14) intersect. This is the case if and only if the shortest distance between both underlying curve segments is less than the sum of the pipes' radii. We therefore calculate this minimal distance and compare to the radii.

More formally, it follows from Eq. 14 that

$$V(\Gamma) \cap V(\Delta) \neq \varnothing \;\Leftrightarrow\; \min_{(x,y)\in A} \|\gamma(x) - \delta(y)\| \leq r_c^+ + r_d^+. \tag{15}$$

This motivates the use of the bivariate distance function

$$D_\gamma^\delta : A = [x_0, x_1]\times[y_0, y_1] \to \mathbb{R},\ (x, y) \mapsto \|\gamma(x) - \delta(y)\|^2 \tag{16}$$

which is a real-valued bivariate polynomial in x and y. To find the points $(x, y) \in A$ where D_γ^δ assumes a minimum, all critical points of D_γ^δ are located using the condition

$$\nabla D_\gamma^\delta(x, y) \stackrel{!}{=} 0. \tag{17}$$

This is equivalent to the following system of bivariate polynomials in x and y, to be solved over $A = [x_0, x_1] \times [y_0, y_1]$

$$\begin{aligned} p(x, y) &:= \dot\gamma(x) \cdot (\gamma(x) - \delta(y)) \stackrel{!}{=} 0, \\ q(x, y) &:= \dot\delta(y) \cdot (\gamma(x) - \delta(y)) \stackrel{!}{=} 0, \end{aligned} \tag{18}$$

which is an algebraic problem of the second type (13). If there is a local minimum within A, it will be among the critical point solutions. However, the global minimum may not be (and typically is not) located inside A, but on its boundary. To that end, the boundaries are also searched for critical points, similarly leading to algebraic problems of the first type (12). Finally, all the critical points are joined together with the corners of A in a set of candidate points for the minimal distance C_A. For any pair of local coordinates $(x^\star, y^\star) \in C_A$, the distance function D_γ^δ is evaluated and as soon as $D_\gamma^\delta(x^\star, y^\star) \leq (r_c^+ + r_d^+)^2$ holds true for one of them, an intersection is reported for Γ and Δ.

Malformed Neurites – Local Self-Intersections Another issue arises for each modeling canal surface individually. For the neurite segment c and its neurite canal segment $\mathfrak{C}(\gamma, \rho_c)$, it is intuitively clear that if for some $x \in [x_0, x_1]$ the curvature $\kappa(x)$ of γ becomes very large or, equivalently, the radius of curvature $\kappa(x)^{-1}$ becomes very small, then the canal surface will self-intersect (see Fig. 7b).

As for neurite-neurite intersections, we detect possible local self-intersections (LSI) of the neurite canal surfaces by using the minimal bounding pipe surface which exhibits a local self-intersection if and only if the curvature radius $\kappa(x)^{-1}$ at any position x is less than its radius (Do Carmo and Do Carmo 1976; Patrikalakis and Maekawa 2002; Rossignac 1985).

This characterization of local self-intersections is used to transform the problem into an algebraic problem of the first type (12). To that end, the condition is examined segment-wise (where γ is guaranteed to be polynomial):

$$\begin{aligned} \kappa(x) r_c^+ < 1 \;&\Leftrightarrow\; r_c^+ \frac{\|\dot\gamma(x) \times \ddot\gamma(x)\|}{\|\dot\gamma(x)\|^3} \\ &< 1 \;\Leftrightarrow\; (r_c^+)^2 \|\dot\gamma(x) \times \ddot\gamma(x)\|^2 < \|\dot\gamma(x)\|^6 \end{aligned}$$

Thus by finding the roots of the polynomial

$$(r_c^+)^2 \|\dot\gamma(x) \times \ddot\gamma(x)\|^2 - \|\dot\gamma(x)\|^6, \tag{19}$$

within the appropriate parameter bounds for x, one can easily verify whether there is a local self-intersection in the pipe surface $\mathfrak{P}(\gamma, r_c^+)$.

Malformed Neurites – Global Self-Intersection In addition, there is the possibility of a neurite canal segment intersecting itself, but not due to excessive curvature, see

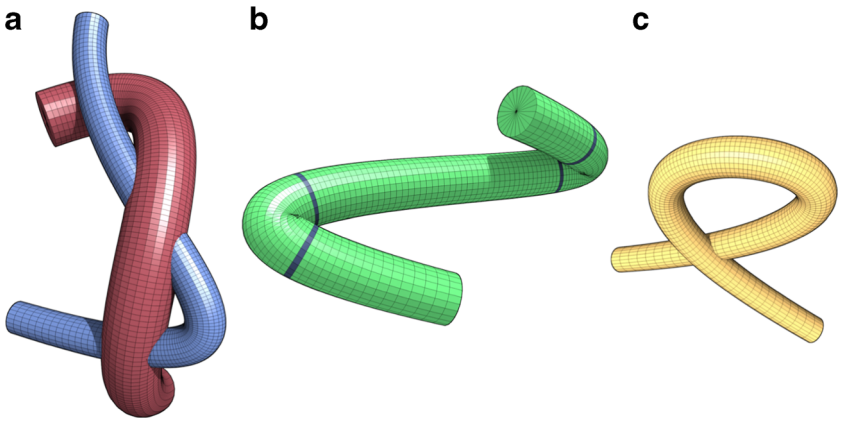

Fig. 7 Examples of intersection between two distinct pipe surface segments (**a**), pipe surface segment local self-intersection (**b**), and pipe surface segment global self-intersection (**c**). Views rendered by a separate canal surface visualization tool

Fig. 7c. Although such "global" self-intersections (GSI) are extremely unlikely given the way that neurite curves are constructed during geometric modeling, they remain technically possible and therefore have to be dealt with in order to give formal guarantees. The detection of such intersections works like that of neurite-neurite intersections, by choosing the segment under examination for both c and d. However, in the initially constructed system, the polynomial $p(x, y) - q(x, y)$ has the non-trivial polynomial divisor $(x - y)$. This divisor has to be factored out to meet the numerical solvers' requirement of a finite solution set, which is achieved as proposed by Maekawa et al. (1998) and Patrikalakis and Maekawa (2002).

Neurite Curve Regularity As a prerequisite for the above analytical formulations, all neurite segment curves $\gamma : [x_0, x_1] \rightarrow \mathbb{R}^3$ are required to be regular and in C^2. The segment curves being polynomial, only their regularity remains to be checked, i.e., $\forall x \in [x_0, x_1] : \dot{\gamma}(x) \neq 0$. This is achieved by locating all real roots of the *regularity polynomial*

$$p(x) = \|\dot{\gamma}(x)\|^2 . \tag{20}$$

Intersection Classification

With the analysis methods above, we have the necessary tools at our hands to detect any possible intersection in the model surfaces. Every model surface segment is tested for regularity of its underlying curve (REG) and local or "global" self-intersections (LSI/GSI). Moreover, every pair of distinct model surface segments is checked for intersections using the neurite-neurite intersection approach. Intersections with soma surfaces are realized as the special case where the soma volume is represented as a pipe surface volume whose underlying curve γ consists exactly of one point (the soma center). The pair-wise intersection computation is facilitated using spatial indexing structures (Octrees) for efficiency. For ease of description later on, we distinguish the two cases:

(i) SONS (soma / neurite segment): intersections between a soma ball and a neurite segment pipe surface,
(ii) NSNS (neurite segment / neurite segment): intersections between two distinct neurite segment pipe surfaces.

While the final surface is to be free of any type of intersection, two special cases of the detected intersection need to be allowed to guarantee its correct connectivity: One is the intersection of neurite root segments with the soma, the other is the intersection between adjacent neurite segments (at the correct respective ends). These intersections are either needed for the "stitching" of individual soma or neurite surfaces during the meshing phase ("The Red-Blue Algorithm") or inconsequential (consecutive neurite segments on the same neurite). We have devised the following set of rules to precisely distinguish required or inconsequential intersections from inadmissible ones:

- LSI/GSI are inadmissible.
- SONS intersections between the soma $s = (1, p_s, r_s)$ and the neurite segment $n = (u, v)$ with segment curve γ are inadmissible unless

  (a) n is a neurite root segment, i.e., there is an edge (s,u) in the morphology tree, and
  (b) the univariate distance function $D^s_\gamma(x) = \|\gamma(x) - p_s\|^2$ for the SONS intersection criterion has exactly one candidate point verifying $D^s_\gamma(x) \leq (r_s + r_n^+)^2$: the corner candidate point corresponding to the neurite root vertex u.

  This exception allows connecting the root neurites to the soma.
- NSNS intersections between neurite segment c with neurite curve γ and neurite segment d with neurite curve δ are inadmissible unless

  (a) c and d share a unique common neurite vertex and
  (b) the bivariate distance function $D^\delta_\gamma(x, y) = \|\gamma(x) - \delta(y)\|^2$ for the NSNS intersection criterion has exactly one candidate point verifying $D^\delta_\gamma(x, y) \leq (r_c^+ + r_d^+)^2$: the corner candidate point corresponding to the common vertex.

  This exception allows connecting neurite surfaces at branching points. Moreover, it disregards all intersections between consecutive segment volumes from the same neurite that occur only due to the half-spheres at both ends of the otherwise tubular segment volume in definition (14).

Definition 3 (Clean morphology tree)
A morphology tree $\mathcal{T}$ is called *clean* if the geometric model computed on $\mathcal{T}$ satisfies all of the following meshing conditions:

(i) There are no minimal distance violations (α-PMDV/β-SMDV).
(ii) There are no regularity violations (REG).
(iii) There are no self-intersections (LSI/GSI).
(iv) There are no inadmissible pairwise intersections (SONS/NSNS).

For a morphology tree $\mathcal{T}$, let the *bounding volume* $B(\mathcal{T})$ be the union of soma volume and the minimal bounding pipe surfaces volumes of all neurite segments. For clean morphology trees, we can make the following observations:

- The absence of GSIs implies that there is no segment pipe surface of non-zero genus.
- If the intersection of every root neurite root segment with the soma is guaranteed, the bounding surface $\partial B(\mathcal{T})$ of the entire morphology is connected.
- The minimal bounding pipe surfaces of neurite root segments are the only modeling surfaces intersecting the soma sphere.
- Intersections between distinct neurite surfaces only occur due to branching points. A fortiori, this excludes intersections between surfaces of two distinct neurite root segments. This condition is biologically plausible, since it can be assumed that two distinct neurites originate from spatially different regions of the soma and that furthermore, once the vicinity of the soma has been left, their plasma membranes do not touch (assuming the absence of gap junctions).
- For a neurite branching vertex $b \in V$, there is exactly one incoming neurite segment $a = (u, b)$ and at least two outgoing neurite segments $o_0 = (b, v_0), \ldots, o_m = (b, v_m),\ m \geq 1$. As there are no forbidden NSNS intersections, only the minimal bounding pipe surfaces for neurite segments in the set $S = \{a, o_0, \ldots, o_m\}$ may intersect around the neurite branching point for b. The classification of NSNS intersections enforces that for every pair $(e_1, e_2) \in S,\ e_1 \neq e_2$, the corresponding two neurite curves γ_{e_1} and γ_{e_2} diverge properly. Hence the surface fragment $\partial\left(\mathcal{P}_a \cup \mathcal{P}_{o_0} \cup \cdots \cup \mathcal{P}_{o_m}\right)$ around the branching point is of genus zero and shaped like a proper tree fork.
- Finally, the definition of allowed NSNS intersections also enforces that consecutive neurite segments $e = (u, v),\ f = (v, w) \in E$ in a neurite path from cannot be joined in a loop, which would create a non-zero genus boundary surface.

These observations imply that if $\mathcal{T}$ is clean, then the bounding volume $B(\mathcal{T})$ has a tree-shaped geometry and its boundary is a connected surface of genus zero.

Mesh Generation

We will briefly outline the terminology used within the documentation for AnaMorph. Let $M = (V, E, F)$ be a triangular polygonal mesh, where V is the (finite) set of vertices, E is the set of edges (containing 2-subsets of V) and $F \subset V^3$ is the set of triangular faces of M. For $e = \{u, v\} \in E$, u and v are called adjacent to one another and both u and v are called incident to e. The graph $G_M = (V, E)$ for M is the undirected graph that has V as the set of graph vertices and E as the set of graph edges.

Faces $f = (v_0, v_1, v_2) \in F$ are ordered 3-tuples of distinct vertices, where all cyclical permutations of f are considered equivalent. This means that for given vertices v_0, v_1, v_2, the tuples (v_0, v_1, v_2), (v_2, v_0, v_1) and (v_1, v_2, v_0) refer to the same triangle, while (v_0, v_1, v_2) and (v_2, v_1, v_0) do not (normal orientation reversed).

The orientations of faces f_1 and f_2 are called compatible with respect to a common edge $e = \{u, v\}$, if the vertices u and v have opposite orders in f and g.

For $v \in V$, the set $N(v)$ of all adjacent vertices is called the *vertex star* or the set of neighbors of v, the set $E(v)$ of all edges v is incident to is called *edge star* of v and the set $F(v)$ of all faces that v is contained in is called *face star* of v. The *dual graph* of M is the undirected graph $G_M^* = (F, E^*)$, where $E^* = \{\{f_1, f_2\} \mid f_1, f_2 \in F \wedge f_1 \text{ and } f_2 \text{ share a common edge}\}$. A vertex $v \in V$ (edge $e \in E$) is called *isolated* vertex, if $F(v) = \varnothing$, *boundary* vertex if the faces in $F(v)$ form a simple path in the dual graph G_M^* (i.e. not a cycle), *regular* vertex if the faces in $F(v)$ form a single cycle in G_M^*. An edge $e \in E$ is called *boundary* edge if it is incident to exactly one face, *regular* edge if it is incident to exactly two faces. $M = (V, E, F)$ is called 2-manifold mesh if all vertices and edges of M are regular. A 2-manifold mesh M is called *orientable* if an orientation on all faces can be chosen in such a way that for all $e \in E$, the two faces sharing e have compatible orientations with respect to e. For an orientable 2-manifold triangular mesh $M = (V, E, F)$, every edge $e = \{u, v\} \in E$ will be assigned the closed line segment $S(e) = \{u + \lambda(v - u) | \ 0 \leq \lambda \leq 1\} \subset \mathbb{R}^3$ and every triangular face $T = \{v_0, v_1, v_2\} \in F$ will be assigned the closed triangle $S(T) = \{\lambda v_0 + \mu v_1 + (1 - \lambda - \mu)v_2 \mid 0 \leq \lambda \leq 1,\ 0 \leq \mu \leq 1\} \subset \mathbb{R}^3$. The *surface* of M shall be defined as $S(M) = \bigcup_{T \in F} S(T) \subset \mathbb{R}^3$. M does not exhibit *self-intersections* if for every pair $T_1 \neq T_2 \in F$ of triangles with $S(T_1) \cap S(T_2) \neq \varnothing$, either (1) T_1 and T_2 share the common edge $e \in E$ and $S(T_1) \cap S(T_2) = S(e)$ or (2) T_2 and T_2 do not share an edge but a common vertex $v \in V$ and $S(T_1) \cap S(T_2) = v$. In the following, orientable 2-manifold meshes that do not exhibit self-intersections will be called *consistent*. Every consistent mesh encloses a volume $V(M) \subset \mathbb{R}^3$ such that $S(M) = \partial\, V(M)$.

Due to the technical nature of the meshing algorithms and the required brevity, many details and rigorous formal proofs cannot be given here. However, the interested reader is referred to Moerschel (2013) for an in-depth theoretical and technical treatment of the entire AnaMorph framework including the meshing process.

The cell meshes for an input morphology $\mathcal{T} = (V, E)$ are constructed in an inductive fashion. The initial mesh is the soma mesh, which is constructed using a recursive tessellation algorithm starting from an icosahedron. Subsequently, one after another, the dendritic and axonal subtrees starting at the soma are created. To that end, the meshing algorithm

starts at the respective root neurite paths and performs the three steps of

(i) meshing the canal surface associated with the neurite path,
(ii) connecting the new surface to the existing cell mesh using the red-blue union algorithm and
(iii) performing the same steps for all neurite paths branching from the current one.

The mesh emerging from this algorithm eventually undergoes a post-processing step improving mesh quality around the "stitching seams" created by the red-blue union algorithm at branching points and the soma.

Meshing Canal Surfaces

For a neurite path P with neurite curve $\gamma_P : [0, 1] \to \mathbb{R}^3$, the mesh M_P for the piecewise defined canal surface of P is generated by carefully sampling the parametric representation (10) on a rectangular parameter grid to produce a regular, high quality geometry containing only triangles with aspect ratios close to one. In order to use the parametric representation (10), it is necessary to suitably define the orthonormal vectors $\mathbf{n}$ and $\mathbf{b}$ appearing in it: For a given *render vector* $\mathbf{r} \in \mathbb{R}^3$ of unit length, we use the *render frame*

$$\mathbf{t}(t) = \frac{\dot{\gamma_P}(t)}{\|\dot{\gamma_P}(t)\|}, \quad \mathbf{n}(t) = \frac{\mathbf{r} \times \mathbf{t}(t)}{\|\mathbf{r} \times \mathbf{t}(t)\|}, \quad \mathbf{b}(t) = \mathbf{t}(t) \times \mathbf{n}(t). \tag{21}$$

Note that $\mathbf{b}(t)$ is the normalized projection of the render vector $\mathbf{r}$ onto the normal plane at $\gamma_P(t)$. Thus the render vector $\mathbf{r}$ acts like a "compass needle" anchoring the render frame in space. The purpose served by this "anchoring" is to ensure that the render frame does not vary too much along γ which would have undesirable twisting effects on the surface mesh (that can be observed, e.g., when using $\mathbf{n}(t) = \|\ddot{\gamma}(t)\|^{-1}\ddot{\gamma}(t)$ instead, the so-called Frenet-Serret framework). However, this orthonormal basis (21) is only well-defined if $\forall t \in [0, 1]$: $\mathbf{r} \times \mathbf{t}(t) \neq 0$, i.e., if the render vector is never tangential to the curve γ. Furthermore, to prevent the render frame from "flipping over", which can result in twisted faces as well, the render vector $\mathbf{r}$ should ideally be chosen "as orthogonal as possible" to $\mathbf{t}(t)$ over $[0, 1]$, meaning that if $\alpha(\mathbf{r}, t)$ denotes the angle between $\mathbf{r}$ and $\mathbf{t}(t)$, then the quantity

$$M(\gamma_P, \mathbf{r}) = \min_{t\in[0,1]} \|\mathbf{r} \times \mathbf{t}(t)\|^2 = \min_{t\in[0,1]} \sin(\alpha(\mathbf{r}, t))^2 \tag{22}$$

should be maximized. Algorithmic access to this problem is again possible through root-finding for univariate polynomial equations. For a constant render vector $\mathbf{r}$, squaring $\|\mathbf{r} \times \mathbf{t}(t)\|$ yields the rational function

$$f_{\mathbf{r}}(t) = \frac{p_{\mathbf{r}}(t)}{q(t)} = \frac{(\mathbf{r} \times \dot{\gamma_P}(t)) \cdot (\mathbf{r} \times \dot{\gamma_P}(t))}{\dot{\gamma_P}(t) \cdot \dot{\gamma_P}(t)} = \sin(\alpha(\mathbf{r}, t))^2. \tag{23}$$

Since γ_P is regular, it follows that $q(t) > 0$, $t \in [0, 1]$ and so $f_{\mathbf{r}}(t)$ does not have any poles. The minimum $M(\gamma_P, \mathbf{r})$ of Eq. 23 for $t \in [0, 1]$ may be found by computing and evaluating all its critical points, which is in turn achieved by differentiation and localization of all real roots over $[0, 1]$:

$$\begin{aligned} \dot{f}(t) &= \frac{\dot{p}_{\mathbf{r}}(t)q(t) - p_{\mathbf{r}}(t)\dot{q}(t)}{q(t)^2} = 0 \\ \iff \quad & \dot{p}_{\mathbf{r}}(t)q(t) - p_{\mathbf{r}}(t)\dot{q}(t) = 0. \end{aligned} \tag{24}$$

To find a suitable render vector, we use an algorithm that generates sets of random direction vectors and selects the best one until it satisfies a predefined quality threshold.

Having thus meshed the canal surface for P, a half-sphere cap is appended to close M_P at its end. If P is a neurite root path, M_P is extended by a cylinder segment between the neurite root point and the center of the soma to ensure a proper intersection with the soma sphere.

The Red-Blue Algorithm

As each neurite surface is meshed individually in the inductive cell meshing process, we require some procedure to merge the individual manifolds into one. This boolean union mesh operation is a basic task in computer-aided design and neighboring research fields and many approaches have been suggested over the years to perform it. Some of them work on parameterized surfaces (Barnhill et al. 1987), others on subdivision surfaces (Biermann et al. 2001) or directly on triangulated surfaces (Lo 1995; Elsheikh and Elsheikh 2014). There are even open source implementations as in The CGAL Project (2016) or Schroeder et al. (2006). As the neurite union problem is a very specific case where we want to be in complete control of what the union algorithm does, we opted for implementing our own algorithm. It is very similar to the one developed in Lo (1995) in that we also construct intersection circles by looking at intersecting edge/face pairs and then remove and remesh the elements at the "seam". For the benefit of anyone intending to use or even enhance AnaMorph, the procedure is described in more detail in the following. We call it the red-blue algorithm.

It takes two consistent meshes $M_R = (V_R, E_R, F_R)$ (the red mesh) and $M_B = (V_B, E_B, F_B)$ (the blue mesh) as input and computes a consistent triangular output mesh M_U that exactly satisfies $S(M_U) = \partial(V(M_R) \cup V(M_B))$. M_U

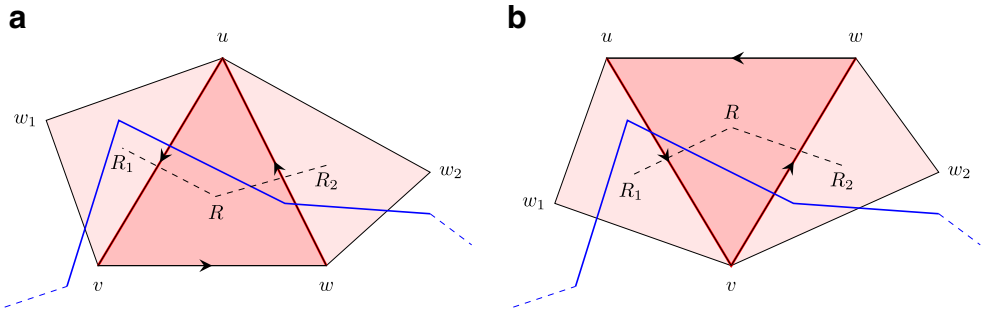

Fig. 8 Simplified two-dimensional projection showing the two possible cases for an affected *red triangle* R: w inside *blue mesh* (**a**), w outside *blue mesh* (**b**). Edges in the dual graph of the *red mesh* are indicated by *black dashed lines*. *Blue lines* represent the boundary of the *blue mesh*

is called the union mesh for M_R and M_B. A red (blue) edge $e \in E_R$ (E_B) is called *simply intersecting*, if the line segment $S(e)$ intersects exactly one blue (red) triangle $S(T), T \in F_B$ (F_R), e is called *complexly intersecting* if it intersects at least two blue (red) triangles and *non-intersecting* otherwise. To exclude edges cases where the union mesh could become non-manifold, we impose the following conditions: Every red (blue) edge $e = \{u, v\}$ intersecting a blue (red) triangle $T = (a, b, c)$ satisfies

(RB1) e intersects T in a single point p_T, i.e. $S(e) \cap S(T) = p_T \in \mathbb{R}^3$.
(RB2) p_T lies properly inside T, i.e. $p_T \in S(T) \setminus \{\{a, b, c\} \cup S(\{a, b\}) \cup S(\{b, c\}) \cup S(\{c, a\})\}$.
(RB3) e properly pierces T, i.e. $p_T \neq u$ and $p_T \neq v$.
(RB4) e is simply intersecting, which can always be guaranteed by suitably splitting e and its incident faces. This process preserves the surfaces $S(M_R)$, $S(M_B)$.

Consider a red simply intersecting edge $e = \{u, v\}$. Using (RB3), we can assume w.l.o.g. that the red vertex u lies properly outside M_B ($u \notin V(M_B)$), whereas v lies properly inside M_B. As M_R is 2-manifold, e is incident to exactly two red triangles R and R_1. Let $R = (u, v, w)$ and $R_1 = (v, u, w_1)$ w.l.o.g., where $w \neq w_1$ are the remaining red vertices from both triangles. Triangles containing an intersecting edge are called *affected*, two affected triangles are called *neighbors* if they share an intersecting edge (simple adjacency in the dual graph is not sufficient here). A simple, yet crucial observation can be made using a parity argument. There are two possible cases for the position of the remaining vertex w of R, see Fig. 8:

(a) w lies inside the blue mesh. Then the edge $\{w, u\}$ is (simply) intersecting and there is a unique red triangle $R_2 = (u, w, w_2) \neq R$ incident to this edge $\{w, u\}$. Clearly, R_2 is an affected neighbor of R. The remaining edge $\{v, w\}$ must be non-intersecting, because both v and w are inside the blue mesh. If the edge $\{v, w\}$ were intersecting, it would have to intersect the surface of the blue mesh an even number $k \geq 2$ of times and would thus be complexly intersecting, contradicting (RB4). Hence R has exactly two intersecting edges and therefore also two distinct affected neighbors.
(b) w lies outside the blue mesh. Then the edge $\{v, w\}$ is (simply) intersecting and it can be seen (using the same argument as in (a)) that the remaining edge $\{w, u\}$ must be non-intersecting. Again, there is a unique second neighbor $R_2 = (w, v, w_2) \neq R$, this time incident to the common edge $\{v, w\}$. Hence also in this case, R has exactly two intersecting edges and two distinct affected neighbors.

The defined neighborhood relation between affected triangles creates an analogous relation for edges: two intersecting red edges e_1 and e_2 are neighbors if they are contained in the same affected triangle R. As seen above, the remaining third edge of R is non-intersecting, and since M_R is 2-manifold, every intersecting edge has exactly two neighbors.

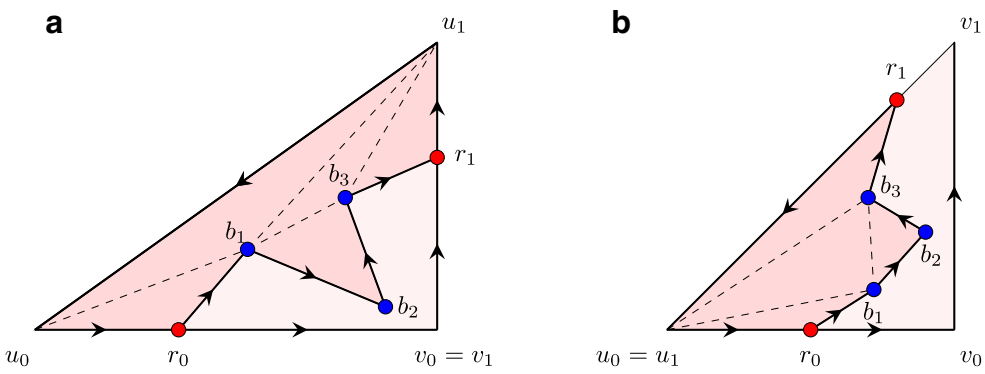

Fig. 9 Two cases for inside and outside polygons defined by $\mathcal{S}$ dividing a *red affected triangle* R: common vertex of the intersecting edges is inside (**a**) or outside (**b**) the *blue mesh*. *Dashed lines* represent possible triangulations of the resulting outside polygon

This property is used in the following construction of a sequence $(R_k)_{k\in\mathbb{N}}$ of affected (red) triangles: Let $R_0 = R$ be an affected triangle and R_1 be one of the two affected neighbors of R. With R_{k-1} and R_k given, where R_{k-1} and R_k are neighbors, let $N \neq R_{k-1}$ be the second remaining affected neighbor of R_k. If N is already contained in the sequence $(R_0, \ldots, R_k)$, the construction is aborted, otherwise the sequence is continued with $R_{k+1} = N$.

Since the red mesh M_R contains a finite number of faces, the above process must terminate. Furthermore, given the initialization R_0 and R_1, it will generate a unique sequence of pairwise different affected triangles $S_R = (R_0, \ldots, R_n)$ of maximal length $n \in \mathbb{N}^0$, where $n \geq 3$. It can easily be seen that the sequence S_R describes a circularly ordered ring of neighboring affected red triangles, whose orientation is determined by the initial choice of the first neighbor R_1.

As a special case, it can happen that for a red triangle ring S_R, the associated blue ring of intersected triangles S_B is trivial, i.e., all red faces in S_R intersect the same blue triangle B and thus there are no blue intersecting edges at all. However, this case can be detected and avoided by suitably subdividing the blue triangle B to create blue intersecting edges. We express this in the additional condition

(RB5) For every matched pair (S_R, S_B) of red/blue triangle rings, both S_R and S_B are non-trivial, i.e., both consist of at least three affected triangles.

Violations of any of the conditions (RB1) to (RB5) are detected. As noted before, (RB4) and (RB5) are asserted by suitable edge and face splits. The unlikely case of violations of (RB1) to (RB3) is treated by re-creating the neurite being connected using a random offset to the angular sampling parameter θ in Eq. 10. Another approach (not implemented at this point) is to add an infinitesimal random displacement to the vertex coordinates. If conditions (RB1) to (RB5) are met, then the rings S_R (S_B) of red (blue) affected triangles form non-trivial cycles in the dual graph G_R^* (G_B^*) of M_R (M_B). In the following, we will assume that there is only one such pair of rings (a condition that can be enforced in the neurite stitching process).

Having computed the pair of affected triangle rings, the intersection polygon $\mathcal{S} = S(M_B) \cap S(M_R) \subset \mathbb{R}^3$ is readily determined. Although all corners of $\mathcal{S}$ lie on the surface of both the red and blue mesh, we distinguish *red corners* (*blue corners*), which are part of a red (blue) intersecting edge and properly lie inside a blue (red) affected triangle. Note that no unaffected red (blue) triangles share a point with $\mathcal{S}$. Let R be an affected red triangle containing the two neighboring intersecting edges $e_0 = \{u_0, v_0\}$ and $e_1 = \{u_1, v_1\}$, where u_0, u_1 (v_0, v_1) denote the red vertices outside (inside) the blue mesh. The edges e_0 and e_1 define two red corners r_0, r_1 of $\mathcal{S}$, in-between which $\mathcal{S}$ shall have $m \geq 0$ blue corners $b_0, \ldots, b_m$ from blue intersecting edges. Again, there are two possible cases, see Fig. 9:

(a) e_0 and e_1 share the common vertex $v_0 = v_1$ inside the blue mesh. With the proper orientation of $\mathcal{S}$, the fragment $(r_0, b_1, \ldots, b_m, r_1)$ divides R into an outside polygon $P_o(R) = (u_0, r_0, b_1, \ldots, b_m, r_1, u_1)$ and an inside polygon $P_i(R) = (r_0, v_0 = v_1, r_1, b_m, b_{m-1}, \ldots, b_1)$. Both polygons are oriented compatibly with R.

(b) e_0 and e_1 share the common vertex $u_0 = u_1$ outside the blue mesh. Again, $\mathcal{S}$ cuts R into an inside and an outside polygon. In this case, these are given by $P_o(R) = (u_0 = u_1, r_0, b_1, \ldots, b_m, r_1)$ and $P_i(R) = (r_0, v_0, v_1, r_1, b_m, b_{m-1}, \ldots, b_1)$, oriented compatibly with R.

In both cases, the outside and inside polygons are planar and simple (i.e., free of self-intersecting sides), although generally non-convex, and thus triangulations are available through standard algorithms for the plane (see also Fig. 9). By symmetry, the same holds for blue affected triangles as well.

The red-blue union algorithm (Algorithm 2) performs a "cutting and stitching" process.

Algorithm 2 Red-blue algorithm for surface mesh union

Input: M_R, M_B: red and blue manifold meshes, intersecting in one polygon.
Output: M_U: the union manifold mesh.

(1) Preparation: Computation of all red and blue affected triangles and an oriented representation of the intersection polygon $\mathcal{S}$, which is achieved robustly and efficiently using Octree lookup and elementary geometric ray/triangle intersection algorithms.
(2) Cutting: Insertion of all corners c of $\mathcal{S}$ as new vertices in M_R and M_B, deletion of all red and blue affected triangles and replacement by triangulations of the inside and outside polygons. Hence $\mathcal{S}$ acts as a seam along which M_R (M_B) is cut into two connected components. Inside / outside polygon triangulation can be performed without data dependency for each affected red (blue) triangle individually, offering maximal parallelization opportunities.
(3) Stitching: Deletion of the inside connected components of M_R and M_B, topological joining of the outside components along their common geometric boundary $\mathcal{S}$.

Provided that the conditions (RB1) to (RB5) are satisfied by both M_R and M_B, it can be shown that the obtained mesh is a consistent union mesh. As an example, Fig. 10a shows

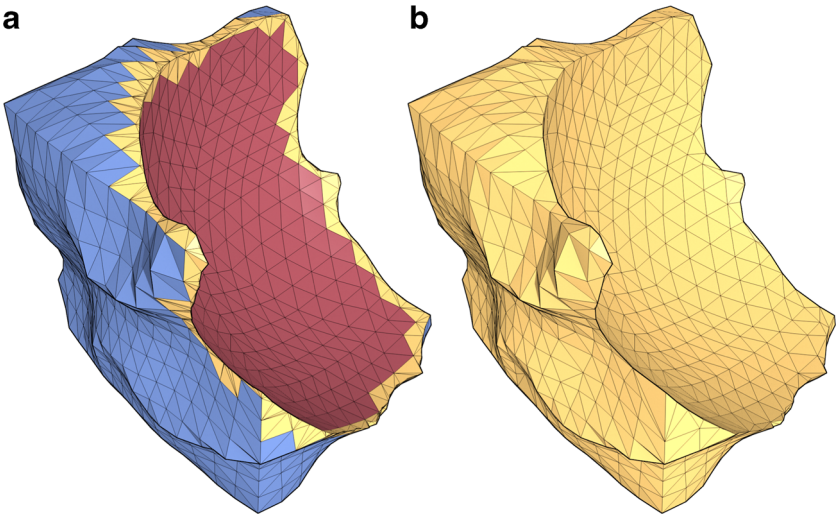

Fig. 10 Set difference mesh generated by the *red-blue* algorithm when carving a *red sphere* out of a *blue distorted cube* (**a**). Mesh obtained after 10% face count reduction using the quadric edge collapse decimation algorithm from Garland and Heckbert (1997) (**b**)

the result of a set difference computation: a red sphere has been "carved out" of a blue distorted cube, resulting in a surface shape with very sharp and thin features, which are difficult to capture in a topologically safe way using isosurface reconstruction algorithms.

Post-Processing

The initial union cell mesh M_U generated in the inductive meshing process generally has an unfavorable triangle structure around the cutting and stitching seams created by the

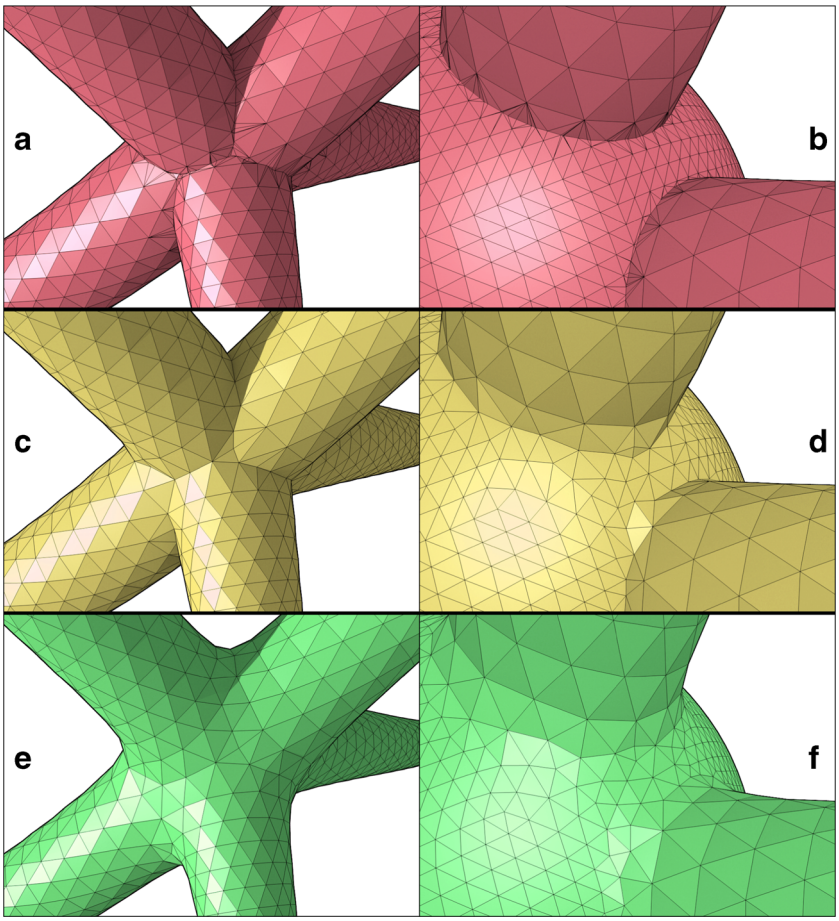

Fig. 11 Results of the post-processing chain around a fictional neurite branching point with four children (**a**, **c**, **e**) and around the connection points of two dendrites to a soma (**b**, **d**, **f**). Union meshes produced by the red-blue algorithm (**a**, **b**), after the greedy edge collapse stage (**c**, **d**), after the surface-preserving smoothing stage (**e**, **f**)

red-blue algorithm (disproportionately small and obtuse triangles with high aspect ratio). To improve mesh quality, we apply a two-stage post-processing chain.

Firstly, we have developed a greedy incremental mesh decimation algorithm performing topologically safe edge-collapse operations to selectively eliminate bad-shaped triangles around the stitching seams. The general structure of this algorithm is similar to other algorithms of this class, e.g., the quadric edge collapse decimation algorithm from Garland and Heckbert (1997). However, instead of performing edge collapses that yield minimum deviation from the original geometry, our greedy algorithm uses information locally available for every triangle to steer the global process in such a way that most triangles are entirely unaffected and the stitching seems are targeted specifically. Informally, every triangle of M_U is assigned a penalty based on its area in comparison to the average area in a certain face neighborhood and its aspect ratio. All triangles are inserted into a max-heap, which is then successively processed by extracting the maximum penalty triangle and, under certain conditions, collapsing it along its shortest edge. After an update of the modified neighborhood, the process continues until only triangles of a certain quality remain. The algorithm is parametrized with different variables to control its behavior and run-time. For a visual example of the edge decimation result, compare (a1) to (a2) and (b1) to (b2) in Fig. 11.

The second stage of the post-processing chain is the surface-preserving HC Laplacian smoothing algorithm presented in Vollmer et al. (1999), applied for a specified number of iterations with suitably chosen parameters. Compare (a2) to (a3) and (b2) to (b3) in Fig. 11 to visualize the effect of smoothing.

The consistency of the created mesh (no holes in the surface, no self-intersections) can be checked using methods of the AnaMorph mesh class.

Table 1 Statistical data for surface meshes before post-processing (UM), after the first post-processing stage (PP GEC) and after both post-processing stages (PP HCS) created from four selected morphologies from NeuroMorpho.Org

Setting	UM	PP GEC	PP HCS	UM	PP GEC	PP HCS
	NMO_04186, McDonald et al. (2007)			NMO_ 09427, Groh et al. (2009)		
PC α	3.52	3.52	3.52	8.67	8.67	8.67
PC β	1.76	1.76	1.76	4.335	4.335	4.335
$\|V\|$	176 584	175 792	175 792	1 864 847	1 859 721	1 859 721
$\|E\|$	529 746	527 370	527 370	5 594 535	5 579 157	5 579 157
$\|F\|$	353 164	351 580	351 580	3 729 690	3 719 438	3 719 438
χ	2	2	2	2	2	2
$A(M)$	7019.8	7018.3	6662.2	21313	21313	20168
$\Delta A(M)$	–	0.0062	5.08	–	0.0018	5.37
$V(M)$	4515.2	4515.2	4250.2	6697.4	6697.3	6204.3
$\Delta V(M)$	–	−0.0002	5.87	–	0.0019	7.36
AR avg	42.3	1.007	1.005	30.7	1.004	1.003
AR std	2.45×10^4	0.036	0.023	5.72×10^4	0.058	0.021
AR max	1.45×10^7	3.05	1.99	1.10×10^8	10.3	3.38
	NMO_00222, Ishizuka et al. (1995)			NMO_ 06189, Vuksic et al. (2008)		
PC α	1.6	1.6	1.6	2.78	2.78	2.78
PC β	0.8	0.8	0.8	1.39	1.39	1.39
$\|V\|$	4641795	4631432	4631432	193394	192593	192593
$\|E\|$	13925379	13894290	13894290	580176	577773	577773
$\|F\|$	9283586	9262860	9262860	386784	385182	385182
χ	2	2	2	2	2	2
$A(M)$	16135	16135	15253	6451.2	6451.0	6116.2
$\Delta A(M)$	–	0.0016	5.47	–	0.0017	5.19
$V(M)$	2058.1	2058.1	1876.4	2822.3	2822.3	2647.5
$\Delta V(M)$	–	0.0016	8.83	–	0.0006	6.19
AR avg	1.21	1.002	1.003	15.6	1.01	1.006
AR std	183.3	0.027	0.017	8436.9	0.049	0.025
AR max	3.98×10^5	5.85	5.00	5.23×10^6	8.16	2.68

Experimental Results

Among hundreds of already meshed morphologies from NeuroMorpho.Org, we have selected four to demonstrate quantitative results about the final output meshes. As for the geometric model, it is hard to convey the final geometry with statistical quantities and illustrations alone, as the visual impression really comes alive only if the geometry can be viewed three-dimensionally. However, statistical quantities serve to illustrate the effects of post-processing and the generally very high mesh quality. Table 1 lists

(1) $|V|, |E|, |F|$ (number of vertices, edges, faces) as well as the Euler characteristic $\chi(M)$,
(2) total surface area $A(M)$ and total enclosed volume $V(M)$,
(3) relative surface area shrinkage $\Delta A(M)$ and volume shrinkage $\Delta V(M)$ (in %) of a post-processed mesh M compared to corresponding original union cell mesh,
(4) aspect ratio of triangles, where for a triangle T with circumradius R and inradius r we use the definition $\mathrm{ar}(T) = R/(2r)$; maximum and average aspect ratio (AR max / AR avg) over all triangles in F as well as the standard deviation (AR std) are also listed.

of the final output mesh $M = (V, E, F)$ produced by the following three settings

(1) **UM** (Union Mesh): the union cell mesh as created during inductive meshing without any post-processing using standard framework settings, while the given parameters PC α and PC β (and $\gamma = \infty$) have been used for the preconditioning.
(2) **PP GEC**: mesh produced after post-processing the result of (UM) with stage one (greedy edge collapse) using the default parameters $d = 15$, $\alpha = 1.75$, $\lambda = 0.125$ and $\mu = 0.5$.
(3) **PP HCS**: mesh produced after post-processing the result mesh of (PP GEC) with the HC smoothing algorithm using the default parameters $\alpha = 0.4$ and $\beta = 0.7$.

All meshes have been verified as consistent using the appropriate methods of the AnaMorph mesh class. A visual impression of the morphology NMO_09427 from Table 1 is given in Fig. 12. Scanning Table 1, it is obvious that already the first stage of the post-processing chain is very successful: average aspect ratios are severely dropped to almost one with very small standard deviations while preserving the surface area and mesh volume almost entirely. This reflects that the bad-shaped triangles around the stitching seams are selectively eliminated, since these cause the unfavorably high aspect ratios in the UM setting while not contributing significantly to mesh volume or area once collapsed. The second post-processing stage (smoothing) further reduces

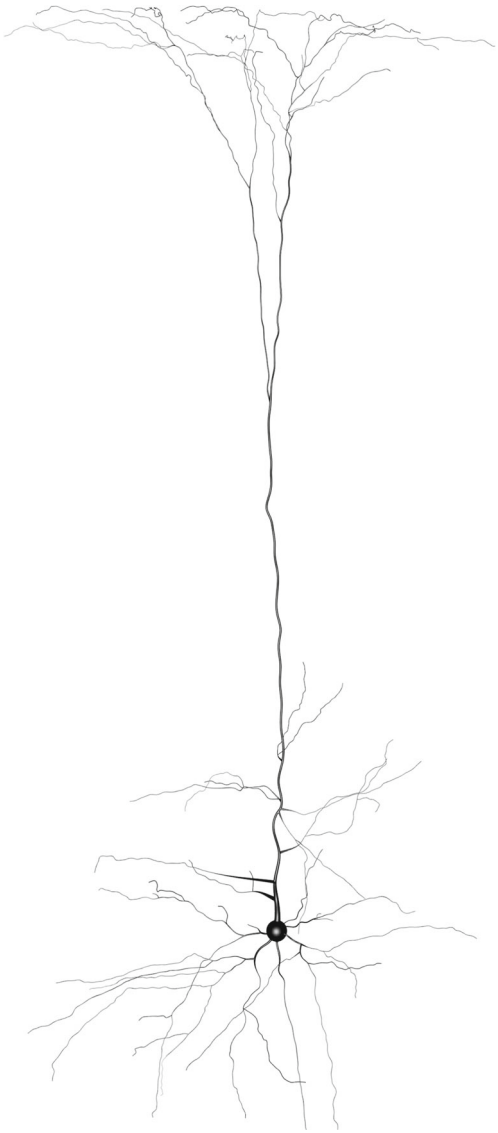

Fig. 12 View of the largest part of cell NMO_09427 (Groh et al. 2009) from NeuroMorpho.Org (statistical quantities in the upper right part of Table 1)

maximum aspect ratio and standard deviation. However, while this increases the usability of the meshes for numerical simulations, the HC algorithm still causes some surface volume shrinkage. These effects can be observed for all morphology meshes, largely independent of factors like cell class or morphological properties. In total, AnaMorph produces final output meshes of consistently very high quality that are suitable as geometry input for advanced numerical simulation methods.

A runtime scaling analysis has been conducted for two scenarios:

(i) Production of a 3d representation for one morphology tree, using varying levels of mesh resolution. This

Table 2 Runtimes (in seconds) of the inductive meshing and post-processing routines for varying levels of surface mesh resolution, expressed in terms of sample points for the angular parameter θ

θ resolution	4	8	16	32	64
faces in resulting obj	27304	92420	347562	1356492	5367680
inductive meshing [s]	0.86	2.6	6.4	23	92
post-processing [s]	0.71	2.7	9.6	38	148

is achieved by setting the angular resolution for sampling of the canal surfaces, i.e., the resolution of the parameter θ from (10).

(ii) Production of 3d representations from morphology trees with varying number of dendrites. To this purpose, we created test geometries with a central soma and N identical dendrites with 500 segments of unit length, spreading radially and in equally spaced angles within the x-y plane.

Runtime measurements for the first analysis are displayed in Table 2. As only inductive meshing and post-processing vary in this analysis, no other runtimes are given. The resolution is doubled from one run to the next and one expects the runtime to increase by a factor of four each time, since the θ resolution also determines the t resolution in (10), which is chosen in such a way that the resulting surface triangles are as equilateral as possible. This complexity can indeed be observed for high resolutions.

Runtime measurements for the second analysis are summarized in Table 3. Except for the preconditioning (which is not needed in the already very regular test morphology trees), all major algorithmic steps have been recorded. All of them scale more or less linearly, with the exception of consistency analysis, which seems to have a runtime complexity of $O(N^2)$ instead of the expected $O(N \log N)$ – obviously, the Octree index is not being put to optimal use yet. This is of no practical consequence, however, since consistency analysis accounts but for a minor fraction of the runtime.

Table 3 Runtimes (in seconds) of the main routines for varying number of identical dendrites

# dendrites	1	2	4	8	16
geometric modeling [s]	0.46	0.87	1.7	3.5	7.0
consistency analysis [s]	0.06	0.16	0.60	2.2	8.5
inductive meshing [s]	0.65	1.6	4.4	9.8	22
post-processing [s]	1.9	3.9	7.8	15.4	31
rest (mainly I/O) [s]	1.03	1.97	4.0	8.1	15.5
total [s]	4.1	8.5	18.5	39	84

Discussion

In this paper, we have developed methods for the automatic generation of three-dimensional computational grids used in numerical simulations of biological processes in cells. Based on a geometric modeling approach and using one-dimensional graph information, cellular morphologies are automatically generated, yielding a numerically high quality surface grid. Tools like the freely available TetGen (Si 2015) can use these to create volume grids required in simulators as UG 4 (Vogel et al. 2013).

We have demonstrated and tested the possibilities of the geometric modeling framework AnaMorph, using one of the largest cell morphology databases NeuroMorpho.Org. Central aspects in the creation of feasible and high-quality meshes are the modeling of neurites as parameterized canal surfaces and the red-blue stitching algorithm. In addition to the automatic geometry generation, we introduced consistency analysis algorithms, allowing detailed evaluation of the quality of the underlying anatomical reconstructions.

The difficulty of constructing a three-dimensional representation of neuronal surfaces from point-diameter data has already received some attention in recent years. McDougal et al. (2013) approach the task by combining constructive solid geometry and surface meshing: Simply put, neurite segments are represented as cone frustum volumes, the neurite volume is defined as the union of all these frusta with spheres at their connection points to fill the gaps resulting from the connection angle. The surface of this volume is then meshed using a modified version of the marching cubes algorithm (Lorensen and Cline 1987). A strength of the marching cubes meshing is that it can easily and efficiently be parallelized to speed up computation. A drawback is that the thinnest neurite in the geometry determines the required resolution of the underlying cartesian voxel grid, which means that regions with bigger diameters (notably, the soma) may be resolved with much finer granularity than necessary, and potentially defining an upper limit for meshable cells. While the authors briefly discuss adaptive grid size as a promising way of circumventing this, another remaining issue is the fact that the marching cubes algorithm creates triangles with greatly varying aspect ratios. This is undesirable if the surface defines the boundary of an unstructured mesh for finite volume discretization, as the robustness of such a scheme depends on reasonably bounded aspect ratios of its elements. The approach taken in AnaMorph aims at creating surface triangles with smallest possible aspect ratio. The major difference between both approaches is the surface model. The surfaces in McDougal et al. (2013) are constructed from a piecewise linear interpolation whose segments are visually identifiable in many places of the output mesh. While AnaMorph's

Table 4 Notation and abbreviations

symbol	meaning	where defined
$\mathfrak{C}$	canal surface	Def. 2
Γ, Δ	pipe surfaces	"Intersection Analysis"
D_γ^δ	distance function	Eq. 16
E	set of edges	
F	set of faces	
κ	curvature function	right after Eq. 11
$l(e)$	edge length	beginning of "The AnaMorph Framework"
M, M_R, M_B, M_U	meshes	beginning of "Mesh Generation"
$P(v_0, \ldots, v_n)$	neurite path	beginning of "Cell Partitioning"
$\mathfrak{P}$	pipe surface	Def. 2
r_c^+, r_d^+, r_n^+	radii of minimal bounding pipes	"Consistency Analysis"
$R_r(e)$	edge radius ratio	beginning of "The AnaMorph Framework"
$S(M)$	mesh surface	beginning of "Mesh Generation"
$\mathcal{S}$	intersection polygon	"The Red-Blue Algorithm"
$\mathcal{T}$	morphology tree	beginning of "The AnaMorph Framework"
V	set of vertices	
$V(\mathfrak{P})$	pipe volume	Eq. 14
AR	aspect ratio	"Experimental Results"
LSI, GSI	local/global self-intersection	"Intersection Analysis"
MDV, PMDV, SMDV	(prim./sec.) min. distance violation	Def. 1
NSNS	neurite-neurite intersection	"Intersection Analysis"
REG	curve regularity violation	"Intersection Analysis"
SONS	soma-neurite intersection	"Intersection Analysis"

non-linear spline interpolation model has an advantage in representing neurites in a physically more detailed way, the meshing technique – sampling the parametric canal surface – requires local regularization of the input data to avoid self-intersections, which is not necessary in the McDougal et al. approach. A combination of the AnaMorph model and a variant of the meshing with more aspect ratio control (like Nielson 2004) might be an interesting prospect for future work on the problem.

Another approach is taken by Lasserre et al. (2012). Although the main purpose of their surface construction algorithm is visualization (of the neurons themselves as well as of numerical simulations on the morphology trees), it may also be used to generate geometries for three-dimensional simulation. The four-step algorithm consists of branch identification (which is essentially the same as what we call cell partitioning), point resampling (which corresponds roughly to AnaMorph's pre-conditioning), extrusion of a piecewise linear coarse mesh along the resulting morphology tree and subsequent surface subdivision for smoothing and refinement. While AnaMorph employs an explicit parameterized model for neurite surfaces, Lasserre et al. use an implicit surface description given by the limit surface of the subdivision process. An interesting feature of the Lasserre et al. method is its multi-level character, which might be very useful for efficient numerical simulation using geometric multigrid solvers. What both the McDougal et al. and Lasserre et al. methods lack is the consistency analysis implemented in AnaMorph. While this should be possible in the former approach by checking for intersections of the generated solid geometries, it seems more difficult to realize in the latter. Avoiding self-intersections is probably non-essential for the visualization purpose – it is very essential, however, for numerical simulation, which is why the consistency analysis is a key feature in AnaMorph.

An issue of some importance in the recreation of realistic morphologies from point-radius data is the representation of the soma or, more generally speaking, of any structure that is not at least approximately radially symmetric (this includes, e.g., dendritic spines). Several different approaches are being used to preserve spatial information for somata:

(a) a single contour line of the soma, typically circumscribing its projection into the xy plane,
(b) multiple cylinders, i.e., treatment of the soma no different from dendrites and axon,

(c) multiple contours that represent the outlines of the soma in parallel slices.

Only the last approach preserves enough information to reconstruct the true spatial shape of the soma. AnaMorph could be modified to use this information in the construction of the soma meshes. Care would have to be taken during the stitching process to ensure that dendrites (and axon) emerging from the soma really intersect the soma surface. Moreover, the consistency analysis would need to deal with the soma description. This can be achieved by using a bounding sphere for the soma for the purpose of the analysis (possibly resulting in a few false intersection reports). Unfortunately, however, according to the information available at NeuroMorpho.Org, only about 5% of the neuron descriptions in the NeuroMorpho.Org database have been generated following the multiple contours approach. The equally rare case of multiple cylinder representation can be dealt with in a straightforward way. The soma can simply be described by a canal surface. As axons and neurites are already treated in the same manner, no additional functionality would have to be implemented. However, the soma would probably have to undergo a special preconditioning since minimal distance violations are to be expected there that must not simply be removed by edge collapses due to the strongly varying radius. Intersection detection might be impaired. The majority of neurons in the NeuroMorpho.Org database (80%) use the single contour approach. A 3d shape cannot be (re-)constructed from this information without further assumptions concerning the extensions in the remaining dimension (e.g., assuming piece-wise sheared cone frustum shape as in McDougal et al. (2013)). Standardization procedures of NeuroMorpho.Org calculate the average position of all points in the contour as the center of a sphere whose radius is defined as the average distance of the contour points from the center. Standardization concludes by transforming the contour vertices to a three-cylinder representation with equal surface as this averaging sphere (presumably to be compatible with popular simulators for point-radius neurons). For the purposes of AnaMorph, we decided to follow this standardization, but use the sphere approximation directly.

An interesting alternative for the construction of realistic somata has been presented by Brito et al. (2013) who use only the positions and extensions of the connections of root neurites to the soma to calculate its surface in an elasto-mechanical model. It appears that many somata can be faithfully reconstructed that way.

As to dendritic spines, if their shape can be approximated sufficiently well by multiple cylinders, AnaMorph can already create their surfaces using the current model. Just like for somata, a more sophisticated preconditioning and intersection detection might be required, however. As AnaMorph is implemented as a C++ library and published under an LGPL license, such a preconditioning (just like any other enhancements on the original code, such as more sophisticated partitioning strategies, export functionality to formats other than wavefront .obj etc.) can be written by anyone reasonably familiar with the C++ language. A fairly extensive in-code documentation in addition to the detailed functional description (Moerschel 2013) will hopefully allow any potential developer to quickly grasp the code structure. New procedures can be used by addition of an option to the command line interface.

AnaMorph continues to be developed and improved. As it has no mesh simplification post-processing step of its own and because the computational cost of numerical applications is usually directly coupled with the size of the underlying grid, one of the most important development efforts aims at creating coarser meshes.

AnaMorph, though applied to cell morphologies in this paper, can be readily used for any lower-dimensional to three-dimensional geometry transformation, where the low-dimensional geometric representation is given as a graph-based "backbone". AnaMorph could, in the future, be used in multiple areas of grid-based numerical simulations. For instance, intracellular organization, e.g., actin filament structures or organelles, like the endoplasmic reticulum which is often conceived as a "cable within a cable" (Shemer et al. 2008), could be processed in similar ways. Another area of application is the cardiac system, which could likely be reconstructed with an AnaMorph-algorithm. At this point, AnaMorph fosters an emergent field of detailed numerical simulation in computation-based neuroscience and virtual medicine. In the future, we intend to identify the critical milestones for integrating AnaMorph with the NeuroMorpho.Org platform. This would offer users the opportunity to directly submit a 3D-reconstruction task from the website and promote the growing demand for full-dimensional simulations.

Notation and abbreviations used in this work summarized in Table 4.

Information Sharing Statement

The AnaMorph code (RRID:SCR_015012) is published (https://github.com/NeuroBox3D/AnaMorph.git).

References

Ascoli, G.A. (2006). Mobilizing the base of neuroscience data: the case of neuronal morphologies. *Nature Reviews Neuroscience*, *7*, 318–324. doi:10.1038/nrn1885.

Barnhill, R., Farin, G., Jordan, M., & Piper, B. (1987). Surface/surface intersection. *Computer Aided Geometric Design*, *4*(1), 3–16.

doi:10.1016/0167-8396(87)90020-3. http://www.sciencedirect.com/science/article/pii/0167839687900203.

Bartoň, M., & Jüttler, B. (2007a). Computing roots of polynomials by quadratic clipping. *Computer Aided Geometric Design*, *24*(3), 125–141. doi:10.1016/j.cagd.2007.01.003.

Bartoň, M., & Jüttler, B. (2007b). Computing roots of systems of polynomials by linear clipping. Technical Report 2007-18, SFB F013 Technical Report.

Biermann, H., Kristjansson, D., & Zorin, D. (2001). Approximate boolean operations on free-form solids. In *Proceedings of the 28th annual conference on computer graphics and interactive techniques, ACM, New York, NY, USA, SIGGRAPH '01* (pp. 185–194). doi:10.1145/383259.383280.

Borg-Graham, L.J. (1999). Models of cortical circuits, Springer US, Boston, MA, chap Interpretations of data and mechanisms for hippocampal pyramidal cell models (pp 19–138). doi:10.1007/978-1-4615-4903-1_2.

Breit, M., Stepniewski, M., Grein, S., Gottmann, P., Reinhardt, L., & Queisser, G. (2016). Anatomically detailed and large-scale simulations studying synapse loss and synchrony using neuroBox. *Frontiers in Neuroanatomy*, *10*, 8. doi:10.3389/fnana.2016.00008.

Brito, J., Mata, S., Bayona, S., Pastor, L., DeFelipe, J., & Benavides Piccione, R. (2013). Neuronize: a tool for building realistic neuronal cell morphologies. *Frontiers in Neuroanatomy*, *7*, 15. doi:10.3389/fnana.2013.00015.

Cignoni, P., Callieri, M., Corsini, M., Dellepiane, M., Ganovelli, F., & Ranzuglia, G.Scarano, V., Chiara, R.D., & Erra, U. (Eds.) (2008). *MeshLab: an open-source mesh processing tool*.

Coombs, J., van der List, D., Wang, G.Y., & Chalupa, L. (2006). Morphological properties of mouse retinal ganglion cells. *Neuroscience*, *140*(1), 123–136. doi:10.1016/j.neuroscience.2006.02.079. http://www.sciencedirect.com/science/article/pii/S0306452206002041.

Do Carmo, M.P., & Do Carmo, M.P. (1976). *Differential geometry of curves and surfaces* Vol. 2. Englewood Cliffs: Prentice-Hall.

Elsheikh, A.H., & Elsheikh, M. (2014). A reliable triangular mesh intersection algorithm and its application in geological modelling. *Engineering with Computers*, *30*(1), 143–157. doi:10.1007/s00366-012-0297-3.

Farouki, R.T. (2012). The Bernstein polynomial basis: a centennial retrospective. *Computer Aided Geometric Design*, *29*(6), 379–419. doi:10.1016/j.cagd.2012.03.001.

Farouki, R.T., & Goodman, T.N.T. (1996). On the optimal stability of the Bernstein basis. *Mathematics of Computation*, *65*, 1553–1566.

Farouki, R.T., & Rajan, V.T. (1988). Algorithms for polynomials in Bernstein form. *Computer Aided Geometric Design*, *5*(1), 1–26. doi:10.1016/0167-8396(88)90016-7.

Floater, M.S., & Surazhsky, T. (2006). Parameterization for curve interpolation. *Studies in Computational Mathematics*, *12*, 39–54.

Garland, M., & Heckbert, P.S. (1997). Surface simplification using quadric error metrics. In *Proceedings of the 24th annual conference on computer graphics and interactive techniques, SIGGRAPH '97* (pp. 209–216). New York, NY, USA: ACM Press/Addison-Wesley Publishing Co. doi:10.1145/258734.258849.

Grein, S., Stepniewski, M., Reiter, S., Knodel, M.M., & Queisser, G. (2014). 1D-3D hybrid modelling – from multi-compartment models to full resolution models in space and time. *Frontiers in Neuroinformatics*, *8*(68), 1–13. doi:10.3389/fninf.2014.00068. http://www.frontiersin.org/neuroinformatics/10.3389/fninf.2014.00068/abstract.

Greiner, H. (1991). A survey on univariate data interpolation and approximation by splines of given shape. *Mathematical and Computer Modelling*, *15*(10), 97–106. doi:10.1016/0895-7177(91)90094-N. http://www.sciencedirect.com/science/article/pii/089571779190094N.

Groh, A., Meyer, H.S., Schmidt, E.F., Heintz, N., Sakmann, B., & Krieger, P. (2009). Cell-type specific properties of pyramidal neurons in neocortex underlying a layout that is modifiable depending on the cortical area. *Cerebral Cortex*, *20*(4), 826. doi:10.1093/cercor/bhp152.

Hines, M.L., & Carnevale, N.T. (1997). The NEURON simulation environment. *Neural computation*, *9*(6), 1179–1209.

Ishizuka, N., Cowan, W.M., & Amaral, D.G. (1995). A quantitative analysis of the dendritic organization of pyramidal cells in the rat hippocampus. *The Journal of Comparative Neurology*, *362*(1), 17–45. doi:10.1002/cne.903620103.

Jüttler, B. (1998). The dual basis functions for the Bernstein polynomials. *Advances in Computational Mathematics*, *8*(1998), S345–352. http://tubiblio.ulb.tu-darmstadt.de/9526/.

Jüttler, B., & Moore, B. (2011). A quadratic clipping step with superquadratic convergence for bivariate polynomial systems. *Mathematics in Computer Science*, *5*(2), 223–235. doi:10.1007/s11786-011-0091-4.

Lasserre, S., Hernando, J., Hill, S., Schuermann, F., de Miguel Anasagasti, P., Jaoude, G.A., & Markram, H. (2012). A neuron membrane mesh representation for visualization of electrophysiological simulations. *IEEE Transactions on Visualization and Computer Graphics*, *18*(2), 214–227. doi:10.1109/TVCG.2011.55.

Levien, R., & Séquin, C.H. (2009). Interpolating splines: which is the fairest of them all? *Computer-Aided Design and Applications*, *6*(1), 91–102.

Liu, L., Zhang, L., Lin, B., & Wang, G. (2009). Fast approach for computing roots of polynomials using cubic clipping. *Computer Aided Geometric Design*, *26*(5), 547–559. doi:10.1016/j.cagd.2009.02.003.

Lo, S.H. (1995). Automatic mesh generation over intersecting surfaces. *International Journal for Numerical Methods in Engineering*, *38*(6), 943–954. doi:10.1002/nme.1620380605.

Lorensen, W.E., & Cline, H.E. (1987). Marching cubes: a high resolution 3d surface construction algorithm. *SIGGRAPH Comput Graph*, *21*(4), 163–169. doi:10.1145/37402.37422.

Maekawa, T., Patrikalakis, N.M., Sakkalis, T., & Yu, G. (1998). Analysis and applications of pipe surfaces. *Computer Aided Geometric Design*, *15*(5), 437–458.

McDonald, C., Eppolito, A., Brielmaier, J., Smith, L., Bergstrom, H., Lawhead, M., & Smith, R. (2007). Evidence for elevated nicotine-induced structural plasticity in nucleus accumbens of adolescent rats. *Brain Research*, *1151*, 211–218. doi:10.1016/j.brainres.2007.03.019. http://www.sciencedirect.com/science/article/pii/S0006899307006051.

McDougal, R.A., Hines, M.L., & Lytton, W.W. (2013). Water-tight membranes from neuronal morphology files. *Journal of Neuroscience Methods*, *220*(2), 167–178. doi:10.1016/j.jneumeth.2013.09.011. http://www.ncbi.nlm.nih.gov/pmc/articles/PMC4197804/.

Moerschel, K. (2013). AnaMorph: a framework for geometric modelling consistency analysis and surface mesh generation of anatomically reconstructed neuron morphologies. Diploma thesis, Goethe-Universität Frankfurt am Main.

Nielson, G.M. (2004). Dual marching cubes. In *Proceedings of the conference on visualization '04, IEEE Computer Society, Washington, DC, USA, VIS '04* (pp. 489–496). doi:10.1109/VISUAL.2004.28.

Patrikalakis, N.M., & Maekawa, T. (2002). Shape interrogation for computer aided design and manufacturing. Springer.

Queisser, G., Wiegert, S., & Bading, H. (2011). Structural dynamics of the cell nucleus: basis for morphology modulation of nuclear calcium signaling and gene transcription. *Nucleus*, *2*(2), 98–104.

Reiter, S. (2012). ProMesh – meshing of unstructured grids in 1, 2, and 3 dimensions. http://promesh3d.com.

Reiter, S. (2014). Effiziente Algorithmen und Datenstrukturen für die Realisierung von adaptiven hierarchischen Gittern auf massiv parallelen Systemen. PhD thesis, Universität Frankfurt am Main.

Rossignac, J.R. (1985). Blending and offsetting solid models (cad/cam, computational geometry, representations, curves, surfaces, approximation). Phd Thesis, The University of Rochester, aAI8528560.

Schroeder, W., Martin, K.M., & Lorensen, W.E. (2006). The visualization toolkit (4th ed.). Kitware.

Schulz, C. (2009). Bézier clipping is quadratically convergent. *Computer Aided Geometric Design*, *26*(1), 61–74. doi:10.1016/j.cagd.2007.12.006. http://www.sciencedirect.com/science/article/pii/S0167839607001434.

Sederberg, T., & Nishita, T. (1990). Curve intersection using Bézier clipping. *Computer-Aided Design*, *22*(9), 538–549. doi:10.1016/0010-4485(90)90039-F. http://www.sciencedirect.com/science/article/pii/001044859090039F.

Shemer, I., Brinne, B., Tegnér, J., & Grillner, S. (2008). Electrotonic signals along intracellular membranes may interconnect dendritic spines and nucleus. *PLOS Computational Biology*, *4*(3), 1–19. doi:10.1371/journal.pcbi.1000036.

Si, H. (2015). TetGen, a Delaunay-based quality tetrahedral mesh generator. *ACM Transactions on Mathematical Software*, *41*(2), 11:1–11:36. doi:10.1145/2629697.

The CGAL Project (2016). CGAL User and reference manual. http://doc.cgal.org/4.9/Manual/packages.html.

Vogel, A., Reiter, S., Rupp, M., Nägel, A., & Wittum, G. (2013). UG 4: a novel flexible software system for simulating PDE based models on high performance computers. *Computing and Visualization in Science*, *16*(4), 165–179. doi:10.1007/s00791-014-0232-9.

Vollmer, J., Mencl, R., & Müller, H. (1999). Improved laplacian smoothing of noisy surface meshes. In *Computer graphics forum* (pp. 131–138).

Vuksic, M., Del Turco, D., Bas Orth, C., Burbach, G.J., Feng, G., Müller, C.M., Schwarzacher, S.W., & Deller, T. (2008). 3D-Reconstruction and functional properties of GFP-positive and GFP-negative granule cells in the fascia dentata of the thy1-GFP mouse. *Hippocampus*, *18*(4), 364–375. doi:10.1002/hipo.20398.

Wittmann, M., Queisser, G., Eder, A., Wiegert, J., Bengtson, C., Hellwig, A., Wittum, G., & Bading, H. (2009). Synaptic activity induces dramatic changes in the geometry of the cell nucleus: interplay between nuclear structure, histone h3 phosphorylation, and nuclear calcium signaling. *The Journal of Neuroscience*, *29*(47), 14,687–14,700.