Interactive Patient-Specific Vascular Modeling with Sweep Surfaces

Jan Kretschmer, Christian Godenschwager, Bernhard Preim, and Marc Stamminger

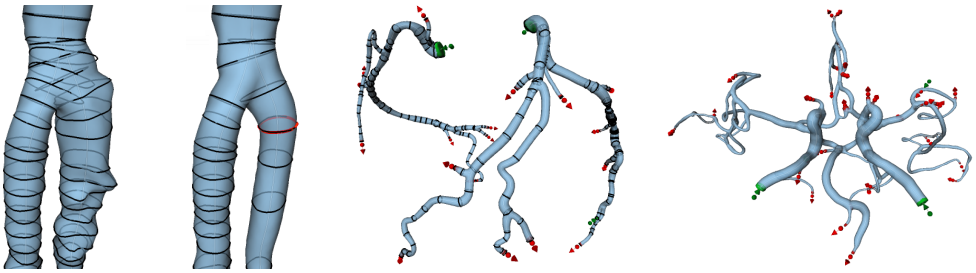

Fig. 1. Incorrect vessel segmentations (*left*) can by modified interactively in high quality by our algorithm (*center left*). An annotation mechanism for inflows and outflows allows the creation of models suitable for computational hemodynamics (*center right, right*)

Abstract—The precise modeling of vascular structures plays a key role in medical imaging applications, such as diagnosis, therapy planning and blood flow simulations. For the simulation of blood flow in particular, high-precision models are required to produce accurate results. It is thus common practice to perform extensive manual data polishing on vascular segmentations prior to simulation. This usually involves a complex tool chain which is highly impractical for clinical on-site application. To close this gap in current blood flow simulation pipelines, we present a novel technique for interactive vascular modeling which is based on implicit sweep surfaces. Our method is able to generate *and* correct smooth high-quality models based on geometric centerline descriptions on the fly. It supports complex vascular free-form contours and consequently allows for an accurate and fast modeling of pathological structures such as aneurysms or stenoses. We extend the concept of implicit sweep surfaces to achieve increased robustness and applicability as required in the medical field. We finally compare our method to existing techniques and provide case studies that confirm its contribution to current simulation pipelines.

Index Terms—Surface modeling, vascular visualization, centerline-based modeling

1 Introduction

Patient-specific models of vascular structures are an important foundation of many clinical applications. Due to their complex morphology and their vital function, vessels are of particular importance when assessing risks or evaluating different surgical strategies. The simulation of blood flows (computational hemodynamics), for instance, is an important new field in medical imaging. The basic idea is to avoid the risks of invasive measurements by using vessel segmentations as a basis to simulate quantifiable patient-specific blood flows. These simulations can then, for instance, be used to compute and visualize wall shear stress gradients, fractional flow reserve (FFR) statistics or intra-aneurysmal flow patterns.

To perform reliable simulations, accurate vascular models are of utmost importance. In pathological cases, such as stenoses or aneurysms in particular, expressive modeling techniques are required to capture the patient-specific vascular geometry. Automatic vessel segmentation is a traditional field in medical imaging and a huge variety of methods has been developed over the last decades. However, none of these methods can guarantee a perfect segmentation in all scenarios. Pathological cases in particular tend to pose problems even for state-of-the-art vessel segmentation algorithms. A manual inspection of automatically generated segmentations is thus usually inevitable before performing computational hemodynamics. In case of deficiencies, a common approach is to run segmentations again with a changed parameter set that hopefully leads to improved results. Another approach is to export data to generic geometric modeling tools where trained profesionals perform plausible corrections [35]. This workflow is impractical for every day's clinical routine as it requires significant geometric modeling skills and involves complex tools.

To solve this problem, we present an image-independent vascular modeling technique. Our method allows to generate high-quality vascular models from automatic segmentations *and* provides facilities to interactively correct those models in an integrated intuitive workflow. We model individual vascular branches using implicit sweep surfaces and smoothly blend them at furcations. To avoid bulging, we use a recently published gradient-based blending operator [23], which perfectly suits our vascular application. We furthermore present an extended interpolation technique that significantly contributes to the robustness of implicit sweep surfaces for challenging datasets. To generate data which is directly suited for simulation frameworks, we propose an automatic inflow and outflow annotation technique which allows for easy user manipulations. We validate our method by evaluating its accuracy on a broad set of medical data sets, ranging from manually annotated coronary arteries to automatically segmented centerline descriptions for whole-body arterial trees. In addition, we compare our method to existing approaches and present case studies, which confirm its contribution to the computational hemodynamics pipeline.

Our method is based on geometric centerline descriptions since they provide a natural way to represent the topology of tubular structures and an excellent basis for navigation during user interactions. In this article, we presuppose the presence of a centerline description for a given vessel tree. The tree may contain automatically segmented cross-sectional contours, as they are generated by geometric segmen-

- *Jan Kretschmer is with the Department of Computer Graphics, FAU Erlangen, and Siemens Healthcare Computed Tomography, Forchheim, Germany. E-mail: jan.kretschmer@cs.fau.de.*
- *Christian Godenschwager is with the Computer Science Department, FAU Erlangen, and Siemens Healthcare Computed Tomography, Forchheim, Germany. E-mail: christian.godenschwager@cs.fau.de.*
- *Bernhard Preim is with the Department of Simulation and Graphics, Otto-von-Guericke University of Magdeburg, Germany. E-mail: bernhard@ovgu.de.*
- *Marc Stamminger is with the Department of Computer Graphics, FAU Erlangen, Germany. E-mail: marc.stamminger@cs.fau.de.*

Manuscript received 31 March 2013; accepted 1 August 2013; posted online 13 October 2013; mailed on 4 October 2013.
For information on obtaining reprints of this article, please send e-mail to: tvcg@computer.org.

tation algorithms [24, 44, 49]. If the centerline tree contains no cross sections, surface models can be created from scratch manually.

2 Related Work

Vascular modeling is a vast field in medical imaging and many different techniques have been proposed to generate models for various applications. In this section we will first review existing vascular modeling techniques and their individual properties. Since our method is based on *implicit sweep surfaces*, we will then give a brief introduction on implicit modeling and a short overview of the history of sweep-based modeling.

2.1 Vascular Modeling

Current vascular modeling methods can roughly be categorized to be either *model-based* or *model-free*. In this section we will first review the most important model-free approaches before giving a brief overview of model-based techniques. Extensive summaries on the topic can be found in [42, 50].

Model-free methods are also referred to as *implicit methods*, since they usually rely on generic point cloud-based interpolation techniques and make extensive use of implicit indicator functions to interpolate models. Common choices for interpolation techniques include multi-level partition of unity implicits [42] and Poisson surface reconstruction [50]. Model-free methods are usually concerned with a robust extraction of point clouds from binary segmentation masks that is able to capture fine vessels. To generate reliable interpolations, these methods need a dense sampling and they do usually not incorporate explicit topological and geometric information on the underlying vasculature.

Model-based methods, in contrast, are motivated by the tubular structure of vascular systems and are frequently used to visualize centerline descriptions. Many techniques rely on explicit mesh generation methods [19, 26, 52], which is usually fast but often leads to self-intersecting meshes at vascular furcations. For computational hemodynamics, the generated models need to be smooth and free of self-intersections or unwanted inner structures. Implicit modeling provides inherent composition mechanisms to solve this problem and has been successfully used to generate model-based vascular models. An early approach [36] was based on convolution surfaces and produces closed, intersection-free models. It has, however, limited expressiveness, since it is restricted to polar definitions of cross sections. An implicit modeling technique that supports free-form shapes was proposed in [30]. The segment-based modeling paradigm, however, leads to C^0 continuous models only, which are inadequate for simulation purposes or virtual angioscopy applications. Sweep surfaces have recently been employed [27] to model vascular trees using piecewise algebraic splines with excellent smoothness properties. The authors, however, report reconstruction times of 1-3 minutes for an optimized GPU implementation, which prohibits interactive applications.

Workflows to interactively segment or correct vascular segmentations have been proposed previously [18, 47]. These semi-automatic methods, however, are bound to the resolution of the underlying dataset and allow only for an indirect manipulation of the vascular representation.

2.2 Implicit Modeling

Implicit indicator functions are a compact natural way to describe the volume and the surface of an object in a scalar field $d(x)\colon \mathbb{R}^3 \to \mathbb{R}$. Even though implicit representations do not contain an inherent parameterization of the volume like polycube-based models [31], their simplicity has made them a widely used tool in geometric modeling. For signed indicator functions, the surface of the object is usually defined by the zero level set $d(x) = 0$, the interior of the object is defined by $d(x) < 0$ and the exterior satisfies $d(x) > 0$

$$d(x) = \begin{cases} < 0 & \text{if} \quad x \quad \textit{inside object} \\ 0 & \text{if} \quad x \quad \textit{on object boundary} \\ > 0 & \text{if} \quad x \quad \textit{outside object} \end{cases} . \qquad (1)$$

Implicit modeling is motivated by the fact that signed indicator functions can easily be combined using Boolean operators such as *min* and *max* to form unions, intersections and subtractions [21, 39]. The result of combining two signed indicator functions is, again, a signed indicator function which allows a recursive application of operators to compose complex objects. This process is commonly referred to as *solid modeling* and is strikingly easy to apply. One drawback, however, is that *min* and *max* operators only preserve C^0 continuity, which leads to undesirable sharp edges at object joints (see Figure 2, left).

To create smooth blends of objects, the closely related concept of *potential field modeling* [10, 12] can be applied. The original concept [10] was physically motivated and focused on point primitives emitting a potential field. In contrast to distance fields, whose magnitudes continuously increases with larger distance to the surface, potential fields are usually designed to drop to zero at a certain distance from the object boundary. A blending operation is required to be neutral if one of the operands is zero. This leads to a locally constrained influence since far-away regions of an object description are neutral during blending. For comprehensive overviews on implicit modeling see Wyvill [51], Bloomenthal [12] and Frisken [21].

An inherent property of potential field modeling is that bulges emerge when overlapping potential fields are blended. In some cases these bulges are desirable (see Figure 2, right) but in general they are considered a problem and several strategies have been proposed to suppress them. One strategy is to offset the blended objects to reduce overlap while maintaining the desired overall shape [11, 22]. This technique, however, was developed for convolution surfaces with circular cross-sectional shapes and reasonable offsets are hard to guess in our case. Another strategy is to narrow the potential field [11, 36]. This, however, results in reduced smoothness properties of the resulting blends. The key observation is that bulges are both a blessing and a curse in potential field modeling and that they have to be treated carefully in an application like vascular modeling.

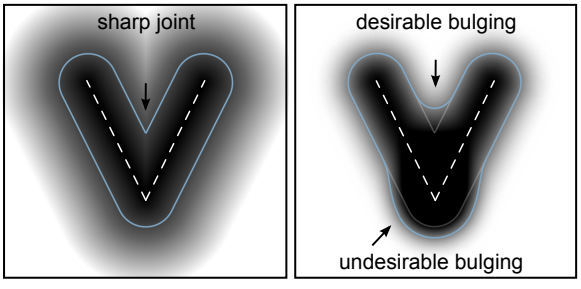

Fig. 2. Comparison of distance field modeling and potential field modeling. *Left:* The distance fields of two line skeletons are unified using *min* (solid union [39]). Sharp joints emerge. *Right:* The corresponding potential fields are blended using summation. Joints are smooth, but bulges emerge (solid union is overlaid as reference).

2.3 Sweep Objects

Sweep objects are generally defined as the shapes which emerge when an object is moved along a trajectory. There are two basic related areas of research: *sweep volumes* treat the movement of three-dimensional objects through space while *sweep surfaces* are commonly used to refer to two-dimensional shapes swept along a trajectory.

Sweep volumes are largely used to describe tool paths in numerically controlled milling [48]. In such applications, Jacobi rank deficiency-based methods [2] and sweep envelope differential equation-based methods [9] are frequently used to compute boundary representations of the resulting sweep. These methods are, however, computationally expensive and require the presence of closed-form boundary representations [1]. Recent work [29] has shown that volume sweeps of polyhedral models can be approximated efficiently by exploiting the fact that in this case only ruled (or developable) surface boundaries are present.

The cross-sectional information in a vascular centerline tree leads to two-dimensional shapes associated with the centerline. Our method can thus be categorized as a sweep surface method. Early appearances

of sweep surfaces [38, 45, 46] described only basic translational, rotational and conic sweeps. The concept was later generalized to curved trajectories with profile curves that vary the size of the contour as it moves along the trajectory [13, 14, 37]. Most of these methods focus on an immediate tessellation of the resulting sweeps in the parameter domain. In this case, the prominent problem is that depending on the trajectory and the shape templates self-intersections frequently occur. Fortunately, implicit surface descriptions allow for volumetric Boolean operations, like unions or intersections, to be performed trivially. This property was exploited to inherently remove any self-intersections by using implicit functions as sweep templates [15, 41]. Crespin and colleagues [15] analytically describe two-dimensional distance fields which can be varied along the trajectory. Their use of an (anisotropic) polar definition, however, restricts the resulting sweeps to star-shaped templates. Schmidt and colleagues [40] further generalized sweep surfaces by generating generic signed distance fields which are blended along the trajectory.

2.4 Implicit Representations for Polygons

To create implicit sweep surfaces from geometric centerline descriptions, some method is required to convert consecutive point lists sampled from the vascular wall into signed implicit descriptions. A well-known approach is variational interpolation [43]. By solving a linear system of equations, coefficients for a set of radial basis functions are computed, which interpolate prescribed values at certain constrained positions. To achieve an interpolation, which is negative inside S_i and positive outside of S_i, additional *interior* and *exterior* constraints with negative and positive interpolation values respectively have to be placed [43]. This is commonly done at a small offset of the interpolation constraints in a direction normal to the surface. While this method is broadly applied as it is straightforward to implement and proven to deliver smooth results, it has several drawbacks. Depending on the support of the chosen basis functions a more or less dense linear system has to be solved for every outline in the vascular model. When dealing with complex outlines, the placement of the interior and exterior constraints has a strong effect on the resulting interpolation. Disadvantageously placed constraints may even result in unusable interpolations. Additionally, the interpolated scalar field massively differs from a signed distance function if it is only constrained in the vicinity of the object's surface. In [40] a hybrid approach is presented, which uses an unsigned Euclidean distance transform [20], to generate additional constraints for the variational interpolation setup. This stabilizes the variational solution and increases the similarity to a signed distance function, but comes at the cost of additional constraints.

Another approach is to construct 2D piecewise algebraic splines [33] which have been successfully used to model vascular trees [27]. This method directly incorporates a smoothness parameter σ and directly produces blendable potential fields of arbitrary continuity. For increased smoothness parameters, however, the potential fields become approximating and do not interpolate the original polygon. In our scenario with closed outlines, this approximating behavior leads to a noticeable shrinkage. Moreover the evaluation of the implicit piecewise algebraic splines is rather expensive even for quadratic splines.

3 Sweep Surfaces for Vascular Modeling

Implicit sweep primitives are defined by a sweep template and a trajectory along which the template is moved. In our scenario the sweep trajectory corresponds to the centerline of an unbranched vascular segment and is described by a list of consecutive nodes F_i which follow the center of the vessel. The corresponding implicit vascular contours O_i are represented by lists of consecutive coplanar points $(o_i^0 \dots o_i^n)$ which sample the vascular surface in the corresponding plane P_i and define a closed polygon.

To create an implicit description of a vascular tree, we essentially perform three steps: first we need to convert all vascular contours O_i into implicit 2D sweep templates T_i (1). We then use the centerline to compute the sweep surface for each branch (2). Finally, the branches are smoothly blended together to form the whole vascular tree (3).

3.1 Implicit Sweep Templates

Several methods have been proposed to generate implicit descriptions from polygons. Since our method is specifically designed to allow interactive real-time applications, we chose to precompute rasterized implicit representations. During evaluation of the implicit sweep, the precomputed images serve as implicit sweep templates, which means, distance computations in the template domain are simple and efficient image lookups. This can be viewed as a caching strategy as opposed to the in-place evaluations used in [27] which result in a significantly higher computational complexity. To reduce aliasing, we use standard bilinear filtering when accessing the template images.

Catmull-Rom splines (CR splines) [17] constitute a popular family of hermite cubic interpolants. We will employ CR splines for several interpolation tasks in this article since they are very easy to construct and expose local control and C^1 continuity. A CR spline that interpolates a list of consecutive points $p_i, i = 0 \dots n$ with varying distances is defined as

$$\begin{aligned} CR_{p_i}^{\tau}(t) &= \begin{pmatrix} 1 \\ u \\ u^2 \\ u^3 \end{pmatrix}^T \begin{pmatrix} 0 & 1 & 0 & 0 \\ -\tau & 0 & \tau & 0 \\ 2\tau & \tau-3 & 3-2\tau & -\tau \\ -\tau & 2-\tau & \tau-2 & \tau \end{pmatrix} \begin{pmatrix} p_{i-1} \\ p_i \\ p_{i+1} \\ p_{i+2} \end{pmatrix} \\ u &= \frac{t-t_i}{t_{i+1}-t_i} \end{aligned} \tag{2}$$

where $t_i = t_{i-1} + |p_i - p_{i-1}|$ and $t_0 = 0$ represent the parameter values of the spline associated with the interpolated points p_i. For every interval $[t_i, t_{i+1}]$, the spline is defined by a local cubic Bezier curve starting at p_i and ending at p_{i+1}. The user-adjustable parameter $\tau \in [0,1]$ accounts for the so-called 'tension' which means it determines how 'curved' the spline is.

To efficiently generate shape templates, we first construct a closed CR spline $CR_{O_i}^{0.7}(t)$ that interpolates the contour samples O_i. To convert this parametric curve to an implicit description, we adaptively subdivide the spline and rasterize the resulting polygon to an image of adjustable resolution. We then perform an unsigned distance transform algorithm [4] which results in a discrete distance field. To convert it to a signed distance field, we use a modified scan line algorithm that uses the subdivided polygon to flip the sign of the distance values inside the object. This is essentially a point-in-polygon test. This method (see Figure 3), is very fast and the quality of the resulting distance fields is mainly determined by the choice of the distance transform algorithm and the resolution of the rasterization. The discrete distance templates

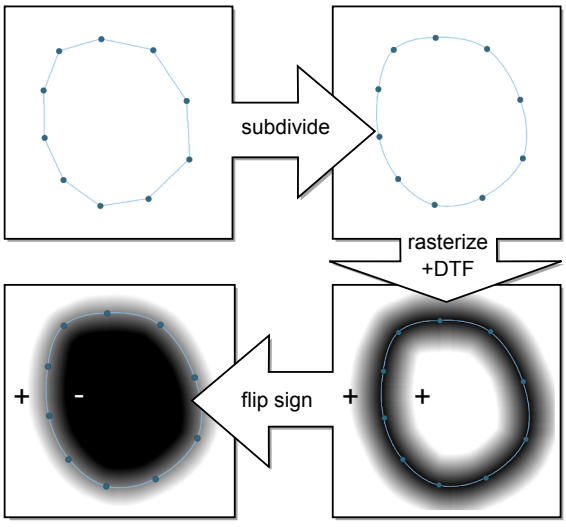

Fig. 3. Diagram of our implicit sweep template generation method. Polygons are converted to Catmull-Rom splines, and rasterized after subdivision. A discrete distance transform algorithm is performed to generate an unsigned distance image. Finally the inside of the polygon is flipped, yielding a signed distance field.

generated by this approach are clamped, but since we are mostly interested in the surface and in the interior of the final sweep, far-away regions of the distance field are irrelevant. To increase performance, the generation of the templates can easily be parallelized.

3.2 Defining the Implicit Sweep

In this section we will describe how to compute an implicit indicator function $B(P)$ from a centerline trajectory with associated sweep templates. The surface of a branch will be described by the zero level set $B(P) = 0$ and the interior of the vessel by negative regions with $B(p) < 0$. Let $F\colon \mathbb{R} \to \mathbb{R}^3$ be a smooth parametric curve that interpolates the sampled centerline positions F_i at parameter values t_i

$$F(t) = (f_x(t), f_y(t), f_z(t)). \tag{3}$$

To define the vascular sweep surface, an affine mapping $W\colon \mathbb{R}^3 \to \mathbb{R}^3$ is needed that transforms positions from the parameter domain of the curve to the embedding space and vice versa. We will from now on refer to the setting in the parameter domain as *tangent space* and denote the embedding space as *world space* as depicted in Figure 4. A

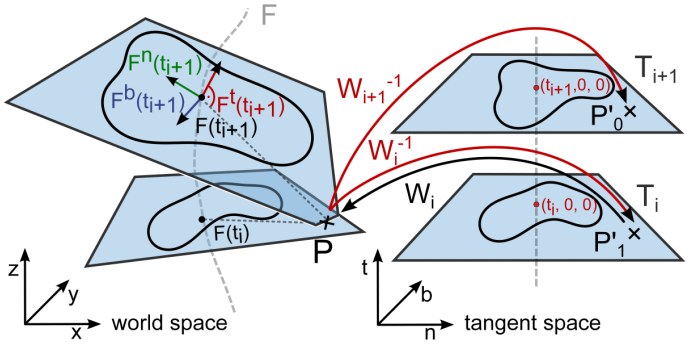

Fig. 4. Illustration of the corresponding world space and tangent space settings of a sweep. The mapping W is defined by the orthonormal frames (F^t, F^n, F^b). For curved trajectories, W^{-1} is ambiguous since world space positions can be mapped to multiple tangent space positions.

common way to define the tangent space is to use Frenet frames which define an orthonormal basis for any curve parameter t

$$\begin{aligned} F^t(t) &= \frac{F'(t)}{\|F'(t)\|} \\ F^n(t) &= \frac{F''(t)}{\|F''(t)\|} \\ F^b(t) &= F^t(t) \times F^n(t) \end{aligned} \tag{4}$$

where F^t is the tangent function and the normal function F^n and the binormal function F^b parameterize a plane which is orthogonal to the centerline. To overcome singularities at inflection points $F''(t) = 0$ several methods have been proposed [8]. The basis defined by F^t, F^n and F^b allows to define the mapping W which transforms tangent space points $P' = (P_t, P_n, P_b)$ to world space points $P = (P_x, P_y, P_z)$ for any given curve parameter t.

$$(P_x, P_y, P_z) = W_t(P_t, P_n, P_b) = F(P_t) + P_n * F^n(P_t) + P_b * F^b(P_t) \tag{5}$$

To describe curved centerlines, we define $F(t)$ using a CR spline $CR_{F_i}^{0.7}(t)$ which interpolates the discrete centerline positions F_i. Unfortunately, due to the bending induced by curved centerlines, the reverse mapping W^{-1} from world space to tangent space is ambiguous. Any world space position may be mapped to multiple positions in the tangent space as depicted in Figure 4. Intuitively, this means, that depending on the curvature of the trajectory and the extent of the templates, local self-intersections may occur. Since our sweep is defined implicitly, however, we can resolve these intersections in-place by successively applying a Boolean union operator. For this, we use the quadratic polynomial operator described in [32] which is a smooth generalization of the *min* [39] operator and preserves the C^1 continuity of our CR spline-based interpolations.

$$min_{li}(x,y) = \frac{1}{2}(x + y - \widehat{|x-y|}_{2,\sigma}) \tag{6}$$

where $\widehat{|x|}_{2,\sigma}$ is a smooth quadric approximation of the absolute function $|x|$ with blend range σ

$$\widehat{|x|}_{2,\sigma} = \begin{cases} |x|, & |x| > 2 \\ x^2(1 - \frac{1}{4}|x|), & |x| \le 2 \end{cases} \tag{7}$$

To prevent an unwanted self-blending for a branch, we have to choose the blend range of the operator to be very narrow. In practice, it should be chosen to be a fraction of the smallest vessel radius, that is present in the dataset. This means min_{li} behaves much like a solid union operator, but has the theoretical property of preserving C^1 continuity.

To determine W^{-1} for a point P in world space, all parameter values t that allow a projection of P to $F(t)$ in the plane defined by $F^t(t)$, have to be determined. The reverse mapping W^{-1} thus looks as follows

$$W^{-1}(P_x, P_y, P_z) = \left\{ W_t^{-1}(P_x, P_y, P_z) \,\middle|\, ((P_x, P_y, P_z) - F(t)) \cdot F^t(t) = 0 \right\}. \tag{8}$$

The computation of the projectable parameters t depends on the degrees of the polynomials used to describe F and F^t. For cubic polynomial interpolations, as in our case, the roots of a polynomial of degree 5 have to be computed numerically [28]. This root-finding has to be performed individually for each local Bezier curve segment of the CR spline. Once the curve parameters t for a query point (P_x, P_y, P_z) in world space have been found, we can map that point to all corresponding positions in tangent space, evaluate the implicit sweep, and compose the implicit values $B'(P_t, P_n, P_b)$ obtained in tangent space using the union operator defined in Equation 3.2

$$B(P_x, P_y, P_z) = min_{li}\{B'(W^{-1}(P_x, P_y, P_z))\}. \tag{9}$$

Sweep templates are usually sparsely present at specific curve parameters t, hence some interpolation method is required to compute the sweep at intermediary positions. For tangent space points, the P_t coordinate directly corresponds to the curve parameter t which leads to a stacked arrangement of the sweep templates (see Figure 4, right). This setting allows for a straightforward interpolation between the implicit templates. In [40], linear interpolation is applied,

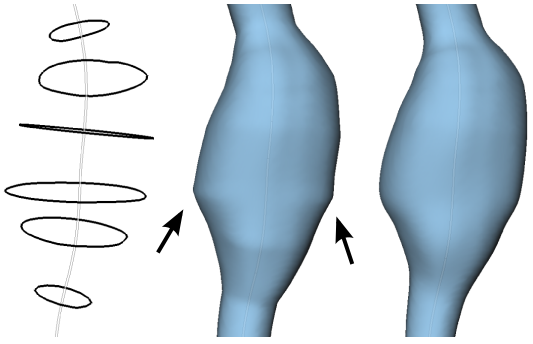

Fig. 5. Comparison of linear sweep template interpolation and Catmull-Rom spline-based sweep template interpolation. *Left:* Original contours. *Center:* Linear interpolations of shape templates lead to visible discontinuities, particularly at high gradients of the template size. *Right:* Cubic Bezier splines produce more organic C^1 continuous results.

which leads to C^1 discontinuities that are undesirable in our scenario

since we want to model organic structures (see Figure 5, left). To blend the distance field templates, we use CR splines $CR_{T_i}^{0.7}(t)$ where $T_i = T_i(P_n, P_b)$ are the values sampled from the implicit sweep template associated with parameter value t_i. The implicit distance value for a point $P = (P_t, P_n, P_b)$ in tangent space is thus defined as follows

$$B'(P_t, P_n, P_b) = CR_{T_i}^{0.7}(P_t). \quad (10)$$

Figure 5 (right) demonstrates the more organic behavior of the cubic template interpolations for a segmented aneurysm.

3.3 Robust Template Blending

Interpolations of signed distance fields generally have to be treated with care. Problems occur if the (negative) inside-regions of adjacent sweep templates are not aligned. Suppose an extreme case, where the inside-regions of two adjacent templates have no overlap at all, as depicted in Figure 6 (left). If the fields are interpolated, the resulting volume will have two disconnected regions, since there are intermediate blends where the whole interpolated template is positive. In cases where the inside-regions of adjacent templates expose a partial overlap, the volume is usually connected, which means the interpolated templates do contain negative regions. These are, however, significantly smaller than those of the surrounding templates, which leads to visible artifacts that could be misinterpreted as stenoses. For well-behaved centerline trees, this problem should not occur, since the centerline is defined to be in the center of a vessel. However, in the presence of pathologies, large calcifications or noisy datasets of low-resolution, automatic segmentation algorithms frequently deliver inconsistent models with a poor alignment of the vascular contours.

To inherently handle such constellations, we introduce a spline-based blending technique which automatically aligns the shape templates in tangent space. The key idea is, not to blend in the t direction as depicted in Figure 6 (left), but to blend along an additional CR spline $CR_{mean_i}^{0.7}(t)$ as depicted in Figure 6 (right). The spline is constructed to pass through the centers of gravity of the vascular contours in tangent space and basically leads a translational alignment of the blended implicit templates. This technique greatly increases the robustness of our system in handling poorly conditioned segmentations.

4 Blending Vascular Branches

To model whole vascular trees, the individual implicit branches B_i constructed in Section 3.2 need to be unified to form the global implicit indicator function V. To achieve a composition with smooth joints at vascular furcations, we use an extension of the Wyvill field function [40] to map the pseudo distance field B of an individual branch to a potential field.

$$f_{wyvill}(x) = (1-x^2)^3 : \mathbb{R} \to [0,1] \quad (11)$$

In contrast to f_{wyvill}, we design our field function $f_w(x) : \mathbb{R} \to [-1,0]$, generates fields which are negative inside objects. By this we can consistently use *min* operators to compute unions

$$f_w(x) = \begin{cases} -1 & x < -w \\ -f_{wyvill}(\frac{x}{2w} + f_{wyvill}^{-1}(0.5)) & -w <= x <= w \\ 0 & x > w \end{cases}. \quad (12)$$

Here, w allows to adjust how fast f_w drops to -1 inside of objects and to 0 outside of objects which means it represents the blending range of the field. While B has global support and the object boundary is defined by $B = 0$, the potential field $f_w(B)$ has local support and the isocontour $f_w(B) = -0.5$ corresponds to the boundary.

A rich set of blending operators is available that allows to create smooth unions of these fields [6, 7, 11, 32, 39] . When dealing with complex vascular trees, there are frequent constellations where vessels run in close proximity to each other. To prevent an unintentional blending of unrelated adjacent vessels, we exploit the topological information of the centerline tree to compute blend weights $b(t) \in [0,1]$ similar to the blend graphs used in [25]. The blend weights b are designed to equal 1 at vascular furcations and to drop to 0 at a reasonable distance along the centerline. This means, the ability of a branch to blend with others or with itself continuously decreases with a larger distance from furcations.

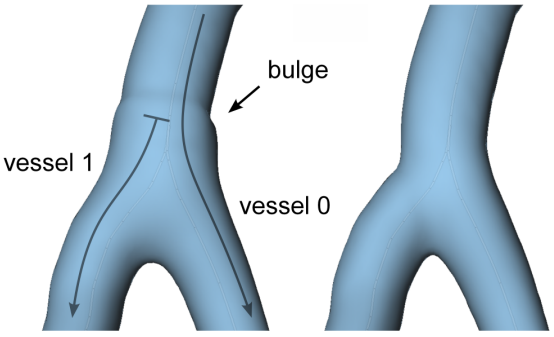

Fig. 7. Comparison of two different blending techniques for composing a vascular tree. *Left* a smooth, but gradient-agnostic blending method [27] leads to visible bulges. *Right:* recent gradient-based blending operators [23] allow for truly bulge-free blending which is perfectly suited for high-quality vascular models.

As discussed in Section 2.2, during composition, undesirable bulges may emerge where potential fields overlap [11, 16]. Figure 7 shows a bifurcation where a child vessel branches off its parent. In our modeling approach, the two vessels have a certain area where they overlap. In this case, the application of standard blending operators leads to a visible bulging artifact that would most certainly influence subsequent simulations on the model. This is also the case for an increased blend range of the operator min_{li} which we used to resolve ambiguities in branches. Fortunately, this problem has very recently been solved [23] by using a quasi-C^∞ operator $min_g(f_1, f_2)$, which allows to generate smooth and truly bulge-free hierarchical blends. The key idea is to modulate the blending properties of the operator with the angle between the gradients of the participating implicit fields. Using min_g, the implicit indicator function V for the whole vascular tree is defined as the recursive binary composition of all individual branches B_i.

5 Interactive Model Correction

Vascular trees are complex structures that usually consist of many branches, each containing arbitrarily many contour definitions. For an interactive correction application, this leads to a demanding environment in terms of user navigation and computational complexity. In this section we will first give a brief overview of our interaction paradigm. We will then describe our culling- and isosurface extraction pipeline that allows an interactive editing of high-quality vascular models.

5.1 Interaction Concept

Our interaction framework uses triangular meshes to visualize the current vascular model during editing. Additionally, a volume dataset can be loaded to verify the segmentation in overlay renderings. Computed tomography angiography (CTA) images provide a contrast agent-enhanced view of the patient-specific vascular structures and thus constitute a common choice. If the user clicks on an arbitrary point on the 3D mesh, a linked orthogonal multiplanar reformation (MPR) view is shown. This allows to inspect the corresponding cross section of the current model as an overlay on the original image (see Figure 11). Using the mouse-wheel, the user is able to move up and down along the current branch which allows for a fast visual inspection of the model. If desired, a snapping feature can be activated which snaps the view to existing nearby contours on the current vessel. If the user is discontent with the model at a certain position, he can simply sketch a corrected contour in the MPR view to get an instantly updated global model. If a prescribed contour was already present, it will be replaced, otherwise a new contour will be inserted into the branch, if a contour is deleted, it will be removed from the model.

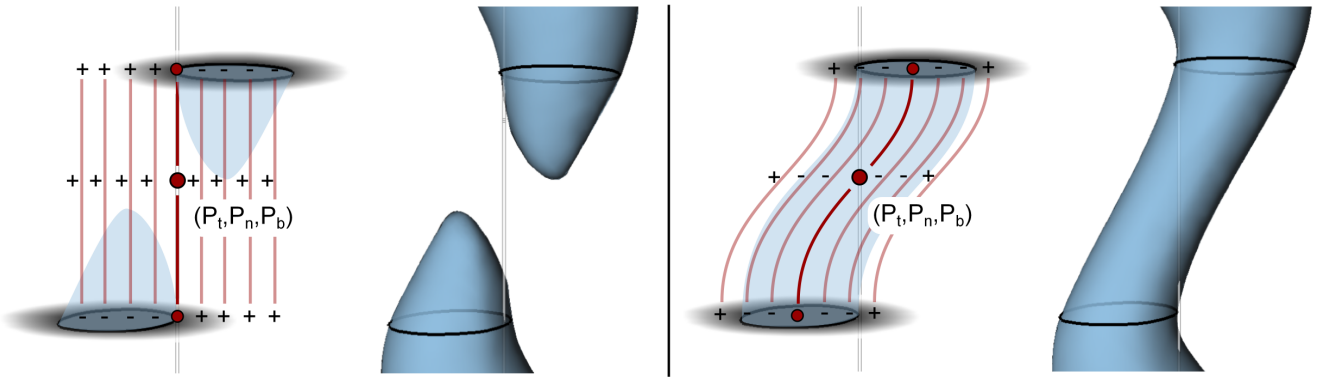

Fig. 6. Illustration of our spline-based interpolation technique in tangent space. *Left:* a poor alignment of adjacent implicit templates may lead to disconnected vessel volumes, since there are intermediate blended templates which are positive (=outside) everywhere. *Right:* by blending along an additional Catmull-Rom spline in tangent space, the templates are implicitly aligned during blending leading to more robust reconstructions.

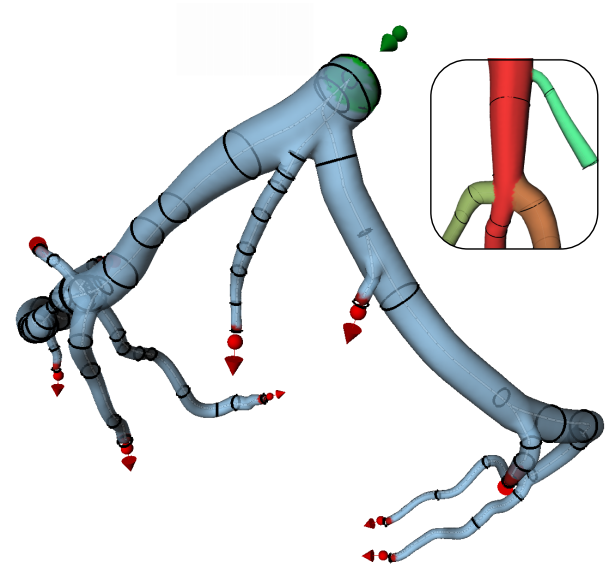

Fig. 8. Based on the centerlines, our modeling tool automatically proposes inflow (green) and outflow (red) tags for a coronary vessel tree (indicated by arrow glyphs). Our global implicit indicator function delivers the corresponding branch and parameter value for any evaluated point P. This allows for an easy assignment of mesh vertices or mask voxels to branches (top, right).

Computational hemodynamics algorithms often require masks or meshes to be annotated with inflow and outflow information. Exploiting the topological information contained in a geometric centerline model, we can automatically determine which end-nodes of the tree will most likely be inflows or outflows. To allow for an easy tagging of mesh vertices or mask voxels, we extended our model evaluation scheme to keep track of the closest vessel and the corresponding parameter value. It is straight forward to do this, since before applying f_w the implicit values of the branches $B_i(P)$ correspond to the distances to the respective branches. By additionally returning the branch with the smallest distance, our system is able to directly associate world space positions P with that branch and the corresponding centerline parameter $F(t)$.With this information we can mark vertices or voxels as inflow or outflow based on their proximity to previously tagged centerline nodes. In interactive sessions, we can thus immediately generate initial proposals for these tags as shown in Figure 8. If areas are tagged incorrectly, they can easily be reassigned by the user using a hot-key.

5.2 Segment-based Culling

To generate a mesh for visualization, polygonizers need to evaluate the implicit sweep V at (many) world space positions $P = (P_x, P_y, P_z)$. For every evaluation of V, we need to compute all potential projections to vessel centerlines (recall Section 3.2). This is the most computationally expensive step in the evaluation process since we need to find the roots of a polynomial of degree five. (i.e. solve for all t in Equation 8). To localize computations, we use a culling strategy based on axis aligned bounding boxes (*AABB*s). For this, we split the CR spline of the centerline into individual piecewise cubic Bezier curve segments for which we compute *AABB*s. Since we have limited a-priori knowledge about the actual surface which belongs to a Bezier segment, we heuristically determine conservative bounds for each segment. Due to the convex hull property of Bezier curves we can use the local control polygon to bound the centerline segment. In addition, we can determine the farthest distance $D_i^{max} = max(|F_i - o_i^j|), j \in 0 \ldots n$ from a centerline position F_i to the corresponding vessel contour O_i. To determine the bounds of a Bezier segment we thus compute the *AABB* of its control points and extend it by the maximum of the distances $max(D_{prev}^{max}, D_{next}^{max})$ as illustrated in Figure 9. Since centerline nodes need not be associated with a contour D_{prev}^{max} and D_{next}^{max} here refer to the closest previous or subsequent node respectively, which is associated with a non empty contour. Since the blending operator g, which we apply as described in Section 4, increases the influence of local segments, the *AABB*s may need an additional extension proportional to the blend range multiplied by the local blend weight $b(t)$. To allow for fast localized queries to V, we finally compute an octree for the segment *AABB*s as depicted in Figure 10. During evaluation we use the octree to gather a local subset of centerline segments which have to be considered when evaluating a point $P = (P_x, P_y, P_z)$.

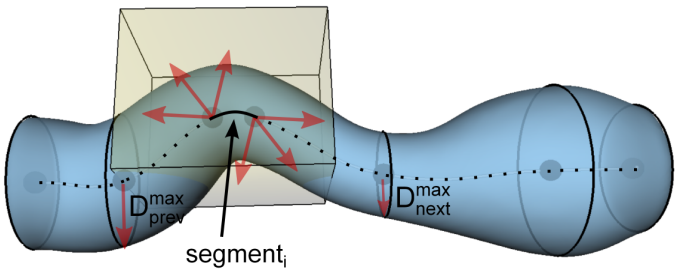

Fig. 9. The maximum distance of the nearest previous and the nearest subsequent contour to the centerline are used to generate a conservative axis-aligned bounding box of the local cubic Bezier segment i.

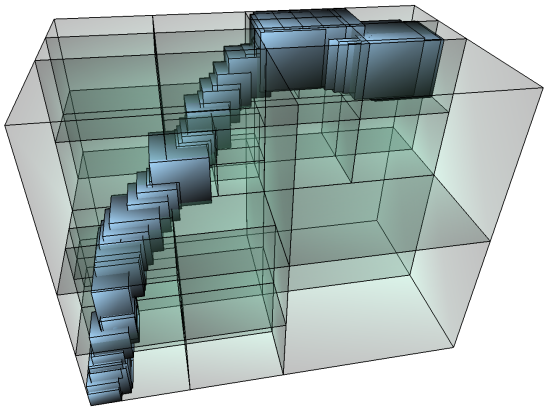

Fig. 10. Illustration of an octree built upon the bounding volumes of the segments belonging to a single branch. During evaluation, only locally relevant segments are considered. Please note, that real world vessel trees are substantially larger than the example in the Figure.

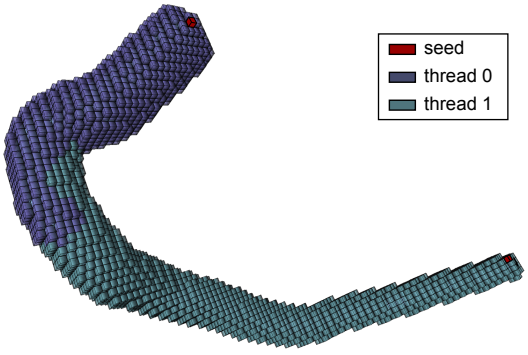

Fig. 12. A multi-threaded propagation-based marching cubes variation is used for interactive isosurface extraction. The marching cubes seed cells (red) are placed at the end points of each branch. The colors of the cells indicate the thread that evaluated the global implicit function. Note that this illustration only shows a single branch.

5.3 Isosurface Extraction

To efficiently generate visualizations during interactive sessions, we use a propagation-based marching cubes (MC) algorithm [5] that extracts mesh approximations of the underlying model. Instead of evaluating all cells in the volume, that method tracks the isosurface with an advancing front from preselected seed cells. For each cell, the selection of neighboring cells which need to be evaluated can be made by examining the marching cubes configuration of the current cell. The problem with such methods usually lies in the determination of the seed cells. In our scenario, however, we can exploit the geometric information provided by the centerline description, since it allows us to make a precise guess about where the surface will pass in space. For each branch, we generate two seeds, one at its beginning and one at its end, yielding $\#branches * 2$ total seed cells. We advance the cell fronts for each seed cell in separate threads which are synchronized at cell access. Figure 12 shows the seeds and cells and their thread affinity for a single vessel branch.

Computational hemodynamics based on meshes has high demands on the triangle quality and density distribution. The polygonization approach described in the previous paragraph does in general not meet these demands. Its purpose lies, however, in the instant visualization of the current implicit model during editing. The final result of our method is the global implicit indicator function V with an arbitrarily high precision that is determined by the resolution of the shape templates T_i. For simulation frameworks that require meshes as input, this indicator function can immediately be polygonized with methods tuned for triangle quality [50].

5.4 Local Model Updates

During user interactions, the aim is to quickly generate updated visualizations. For this, we use a localized update strategy that only recomputes the changed portions of the surface. When the user deletes, modifies or creates a contour, we bound the modification based on the *AABB*s computed in Section 5.2. For this, we need to account for the support of the CR spline-based template interpolation. If an implicit template changes, two neighboring segments may potentially change their appearance. During a modification, the model may grow or shrink. We thus create an *AABB* that bounds all affected segments and update the model description with the new contour. We then create another *AABB* for the modified affected segments and finally merge those two *AABB*s, which yields the final bounds $AABB_{mod}$ for the user modification as indicated in Figure 11 (center, left).

To create an updated surface mesh, it is sufficient to recompute the surface inside of $AABB_{mod}$. For this, we first clear the evaluations of the previous MC run in $AABB_{mod}$. While doing this, we flag all cells which previously intersected the surface and lie at the border of $AABB_{mod}$ as seeds. Since we know the new surface will be consistent with the old surface beyond $AABB_{mod}$, these seeds provide perfect entry points for the subsequent local isosurface extraction. In Figure 11 (center, right), seed cells generated this way are visualized in red color.

6 Results

To validate the models generated by our method, we applied it to a variety of segmentations, including cerebral vessel trees, arterial whole-body vessel trees and coronary vessel trees. For these applications our method generated smooth high-quality models with an instant manual correction capability. In addition, our method has been applied for the manual creation of 30 coronary segmentations in an ongoing validation study that compares computational hemodynamics-based FFR simulations with invasively measured blood flow data.

6.1 Model Smoothness

Since our method uses CR splines to define the sweep templates and trajectories and to interpolate the sweep in tangent space, our models generally expose C^1 continuity. This continuity is preserved by the union operator min_{li} and the gradient-based branch blending operator min_g. By design, our method interpolates all prescribed contours which is important for consistent feedback during interactive editing sessions. For the visualization of vascular trees from noisy segmentations however, the ability to generate smoothed results is important. Our method can easily be extended to support this by applying smoothing kernels to the discretized shape templates T_i. Another way is to blend the templates with a smooth shape, which leads to a modified template function

$$T_i'(P_n, P_b) = T_i(P_n, P_b) * (1 - s) + shape(P_n, P_b) * s.$$

This can be seen as a regularization scheme which allows to restrict

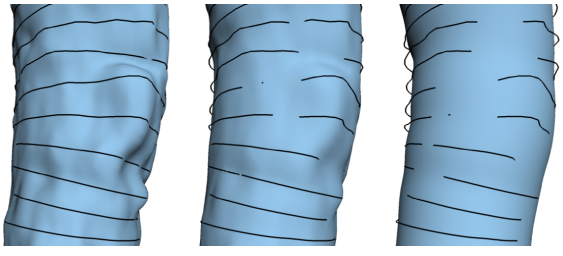

Fig. 13. Noisy centerline models (*left*) can be smoothed by blending the implicit templates of the cross-sectional contours with implicit analytic shapes like circles or ellipses. This leads to a controllable regularization. Smoothness parameters s from left to right are 0, 0.5 and 1.

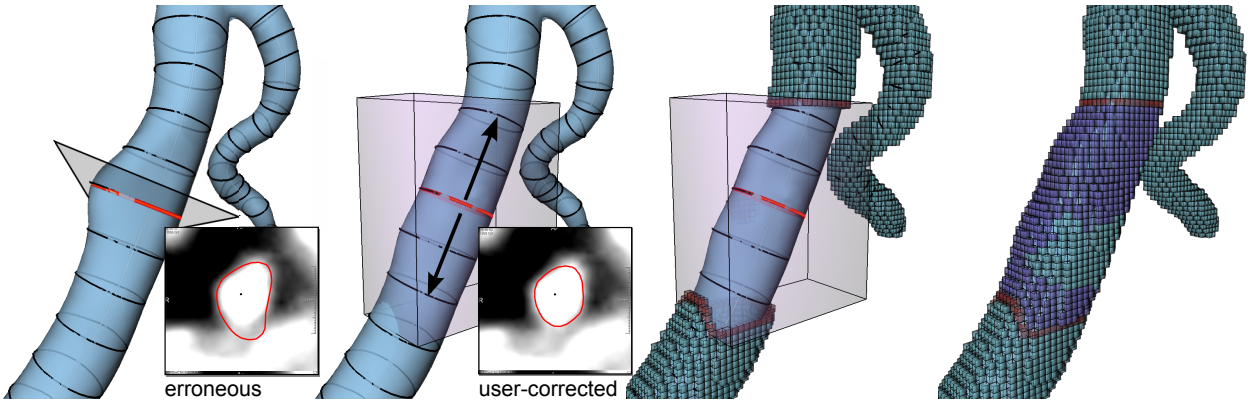

Fig. 11. Illustration of our local modification scheme. *Left:* the automatic contouring result has leaked to a close-by gradient, which is a frequent problem. *Center, left:* the user can select the erroneous contour by clicking in a 3D view and sketch a correction in an MPR view of the image. The region where the surface of the model might change corresponds to the support of the Catmull-Rom spline and is indicated by the arrows. *Center, right:* the bounding box, which encloses the affected segments is cleared from the previous marching cubes state and border cells are used as seed cells (red) for subsequent isosurface extraction. *Right:* the final constellation after local execution of the propagation-based marching cubes.

the cross-sectional shapes of vessels to a limited deviation from a target shape depending on a smoothness parameter $s \in [1,0]$. Typical shapes are ellipses or circles which can directly be fitted to the corresponding contour polygon. Figure 13 shows the impact of an increasing smoothness parameter s.

6.2 Comparative Study

We qualitatively compared our method to the technique described in [27], since it is also based on implicit sweep surfaces and to our knowledge the only method with a model quality comparable to ours. The main difference is that the authors use piecewise algebraic splines (PAS) with an inherent smoothness parameter σ as the underlying implicit representation. This results in superior theoretical smoothness and continuity properties for their models. To perform a direct comparison, we implemented the method described in [27] and incorporated it in our system. Figure 14 shows close-up views of a coronary model that illustrates the high similarity of both methods for cubic PAS splines with a low smoothness parameter $\sigma = 2$. The most prominent difference we noticed in the resulting models was a shrinkage of vessels for increased smoothness parameters σ. As mentioned in Section 2.4, this effect is induced by the approximating character of the implicit algebraic spline representation [33] for the vascular contours. An increased smoothness thus comes at the cost of an increased shrinkage as shown in Figure 14 (center, right). Our explicit blending-based smoothing strategy (see Section 6.1) does not suffer from this effect. The geometric regularization shape can even be designed to have the same area as the corresponding free-form contour. This results in a very low impact of smoothing on the overall volume of a model which is an important quantity in the vascular domain.

The development of our method was driven by the observation, that a manual inspection and correction system is essential for practical computational hemodynamics applications. Hence, an important feature of our method is its support for interactive editing. A major drawback of the PAS method is its computational complexity and the resulting model generation times. The reconstruction times of 1-3 minutes reported in [27], which were already achieved with an optimized CUDA implementation, do not allow for interactive workflows.

Since [27] presents a combination of a segmentation algorithm and a subsequent modeling method, the presence of a cross-sectional contour is assumed for every centerline point. Our method explicitly supports centerline points that do not contain a vascular contour, but only contribute information about the centerline trajectory. This allows the sampling of the centerline to be independent of the number of cross-sectional definitions. For a given centerline of arbitrary complexity, a single contour is thus sufficient to define the whole sweep with a constant cross-section. In addition, this allows the user to adaptively place more contours in pathological regions like stenoses or aneurysms, while leaving less detailed regions with sparse contour information.

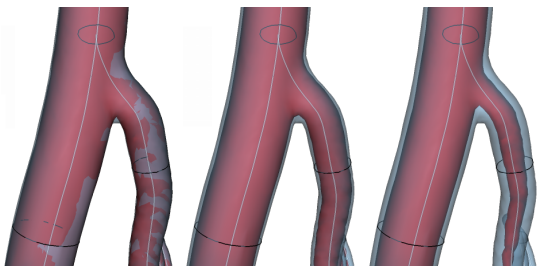

Fig. 14. Qualitative comparison of cubic PAS models [27] (red surface) with our models (blue overlay). Smoothness parameter *sigma* for [27] from left to right: 2, 5, 6. Blend range for our method was set to $w = 3mm$ in all examples. *Left:* for a low σ both methods are qualitatively indistinguishable (notice the z-fighting of the overlays which results from highly similar surfaces). *Center, Right:* for higher smoothness parameters σ PAS models visibly shrink due to their approximating nature.

6.3 Quantitative Comparison

To quantitatively validate our method, we compared the results to our implementation of PAS and an implementation of convolution surfaces (CS) [36] that is shipped with the medical imaging framework MeVisLab [34]. Table 1 shows Hausdorff distance statistics measured between our models and meshes extracted from PAS and CS models. For $\sigma = 2$ the mean distance between our models and PAS models is $< 0.07mm$ which is fraction of the voxel size delivered by current CT or MRI protocols. Note that the high increase in the surface distances to the PAS method for $\sigma = 6$ is due the loss of entire regions of thin vessels during isosurface extraction. This is an effect of shrinkage since the uniform MC algorithm cannot adequately sample vessels if they become too thin. As indicated by Table 1, the effect is particularly pronounced for vascular trees with thin vessels like coronary or cerebral trees.

The cerebral tree model only contained radii associated with centerline nodes. Our method easily supports this type of data, if the shape templates are replaced with implicit circle definitions, similar as in the smoothing approach presented in Section 6.1. Since CS only support radial models, we could not compare the results for the other datasets. CS models and our models are, however, very similar except for the fact that convolution surfaces inherently generate rounded caps at vessel ends. These are precisely the regions where both methods have the largest differences and which account for the *max* error in Table 1.

Table 1. Max, mean and rms Hausdorff distances (in millimeters) computed with MeshLab. Distances compare piecewise algebraic splines (PAS) with different smoothness parameters σ and convolution surfaces (CS) to our results. The arterial and coronary datasets could not be compared to CS since the centerline models contained free-form outlines and CS only supports circular vessel shapes.

		PAS			CS
		$\sigma = 2$	$\sigma = 5$	$\sigma = 6$	
arterial tree	*max*	1.286	0.896	1.182	-
$\sim 90K$ vertices	*mean*	0.074	0.140	0.210	-
(Figure 15)	*rms*	0.102	0.172	0.240	-
coronary tree	*max*	0.516	0.572	15.946	-
$\sim 60K$ vertices	*mean*	0.039	0.128	0.489	-
(Figure 8)	*rms*	0.058	0.138	1.474	-
cerebral tree	*max*	0.199	12.777	26.533	1.643
$\sim 150K$ vertices	*mean*	0.017	0.594	2.0673	0.097
(Figure 15)	*rms*	0.023	1.846	4.396	0.157

6.4 Performance

The time needed to generate and visualize our models is mainly determined by the number of vascular cross sections that are present and by the resolution of the mesh that is extracted. The rasterization time for the cross sections depends on the target resolution of the template images. In all examples shown in this article, this resolution was set to 64×64 since it is a good trade-off between accuracy and rasterization time. Computation times are, however, mostly dominated by the isosurface extraction since it involves the costly root finding as mentioned in Section 5.2. The initial complete mesh generation time is usually in the range of 2.5 - 4.5 seconds for real world vascular trees with a marching cubes resolution of 256^3 (see Table 2). After this, interactive local modifications take less than 0.5 seconds on our datasets which is fast enough for interactive feedback. Local update times are mainly determined by the density and the constellation of the templates. Here, models with a dense template configuration result in faster update rates, because the template support is spatially more restricted. Changing a template in a sparse model, in contrast, causes large portions of a branch model to change which leads to a more global recomputation of the mesh. This behavior is reflected in Table 2 which shows that updates for the cerebral tree are very fast, even if its model is substantially larger than the arterial and the coronary trees.

Table 2. Mesh generation times (in seconds) for three vascular trees (marching cubes with 256^3 cells) taken on an Intel Xeon CPU with 4 physical cores @2.80GHz and 8GB RAM. Timings are for rasterizing the implicit *templates*, for extracting the *mesh* and the average time needed to compute an *updated* mesh after contours have been modified.

	Templates	Mesh	Updates (Avg)
arterial tree (398 cross sections)	0.44	2.06	0.15
coronary tree (109)	0.25	2.49	0.45
cerebral tree (2422)	2.29	2.10	0.11

6.5 Simulation Example

The benefit of our manual correction method can be demonstrated by the following example. The velocity visualizations shown in Figure 16 were generated using a Lattice-Boltzmann simulation technique [3] that was executed on a binary mask generated from our global indicator function V. Figure 16 (left) shows the results for the original automatic centerline-based segmentation [24]. Since it did not accurately capture a present calcification, the velocity magnitude color-coded streamlines exhibit no saliencies and indicate a normal blood flow. The results on the right, in contrast, were generated from a corrected vascular model and clearly indicate the temporarily increased velocity of the blood flow inside the stenosis. This illustrates, that even small shortcomings in a segmentation can result in significantly distorted simulation results.

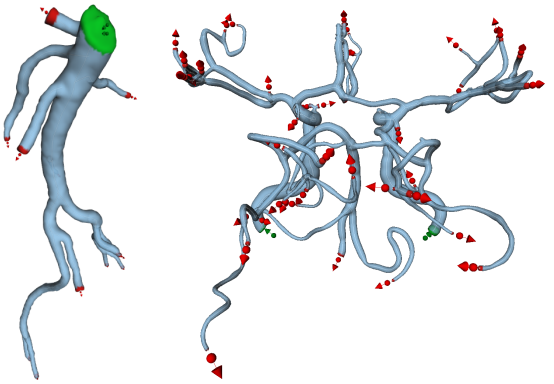

Fig. 15. An arterial model and a cerebral model generated with our method. Both trees received automatic inflow and outflow tagging and can be modified in real-time in our interactive prototype.

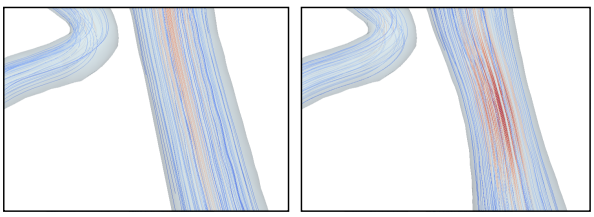

Fig. 16. Streamline visualization with color-coded simulated blood velocities generated with a Lattice-Boltzmann method [3].*Left:* the automatically segmented model did not capture a stenosis that was present and thus leads to an unsuspicious velocity pattern. *Right:* the corrected model reveals the increased blood velocity inside the stenosis.

7 Conclusion and Future Work

We have presented a fast and robust method to generate high-quality vascular models from geometric centerline descriptions. By employing a local update mechanism, it is in particular fast enough to allow for local model updates at interactive rates without resorting to a simplified model representation. Our method incorporates a user-adjustable smoothness parameter which allows to choose between a strictly interpolatory or an approximating behavior depending on the application. A blend range parameter can be used to control the blending behavior at vascular furcations. The implicit modeling approach guarantees that all models are self-intersection free and expose C^1 continuity. In addition it inherently provides an assignment of locations in space to a specific vessel branch and the corresponding parameter of the centerline spline. This property can easily be used to automatically annotate meshes and voxelizations with inflow and outflow information as it is required by many computational hemodynamics frameworks. Our method thus enables the creation efficient workflows to manually inspect and correct automatic segmentations or to create new ones given a presegmented centerline tree. We have applied and evaluated our approach on a variety of clinical datasets and have shown its improvements over similar state-of-the-art methods. We have provided case studies that confirm the important contributions that our method brings to the clinical computational fluid dynamics pipeline.

Currently our method is implemented in a prototyping application. Future work will focus on the tuning of our interaction concepts for the clinical environment and on the tight integration of our method into corresponding simulation frameworks.

REFERENCES

[1] K. Abdel-Malek. Closed-form swept volume of implicit surfaces. In *Proceedings of the 26 th ASME Design Automation Conference*, 2000.

[2] K. Abdel-Malek and H.-J. Yeh. Geometric representation of the swept volume using jacobian rank-deficiency conditions. *Computer-Aided Design*, 29(6):457 – 468, 1997.

[3] C. K. Aidun and J. R. Clausen. Lattice-Boltzmann method for complex flows. *Annual Review of Fluid Mechanics*, 42:439–472, 2010.

[4] M. Akmal Butt and P. Maragos. Optimum design of chamfer distance transforms. *IEEE Transactions on Image Processing*, 7(10):1477–1484, 1998.

[5] C. L. Bajaj, V. Pascucci, and D. R. Schikore. Fast isocontouring for improved interactivity. In *Proceedings of the 1996 symposium on Volume visualization*, pages 39–46, 1996.

[6] L. Barthe, B. Wyvill, and E. De Groot. Controllable binary CSG operators for "soft objects". *International Journal of Shape Modeling*, 10(02):135–154, 2004.

[7] A. Bernhardt, L. Barthe, M.-P. Cani, and B. Wyvill. Implicit blending revisited. In *Computer Graphics Forum*, volume 29, pages 367–375, 2010.

[8] R. L. Bishop. There is more than one way to frame a curve. *American Mathematical Monthly*, pages 246–251, 1975.

[9] D. Blackmore, M. Leu, and L. Wang. The sweep-envelope differential equation algorithm and its application to NC machining verification. *Computer-Aided Design*, 29(9):629 – 637, 1997.

[10] J. F. Blinn. A generalization of algebraic surface drawing. *ACM Transactions on Graphics*, 1(3):235–256, 1982.

[11] J. Bloomenthal. Bulge elimination in implicit surface blends. In *Implicit surfaces*, volume 95, pages 7–20, 1995.

[12] J. Bloomenthal and K. Shoemake. Convolution surfaces. In *ACM SIGGRAPH Computer Graphics*, volume 25, pages 251–256, 1991.

[13] W. Bronsvoort, P. Nieuwenhuizen, and F. Post. Display of profiled sweep objects. *The Visual Computer*, 5:147–157, 1989.

[14] S. Coquillart. A control-point-based sweeping technique. *IEEE Computer Graphics and Applications*, 7(11):36 –45, 1987.

[15] B. Crespin, C. Blanc, and C. Schlick. Implicit sweep objects. *Computer Graphics Forum*, 15(3):165–174, 1996.

[16] E. de Groot, B. Wyvill, and H. van de Wetering. Locally restricted blending of blobtrees. *Computers & Graphics*, 33(6):690–697, 2009.

[17] T. D. DeRose and B. A. Barsky. Geometric continuity, shape parameters, and geometric constructions for catmull-rom splines. *ACM Transactions on Graphics*, 7(1):1–41, 1988.

[18] S. Diepenbrock and T. Ropinski. From imprecise user input to precise vessel segmentations. In *EG Visual Computing for Biology and Medicine*, pages 65–72, 2012.

[19] P. Felkl, R. Wegenkittl, and K. Bühler. Surface models of tube trees. In *Proceedings of Computer Graphics International*, pages 70–77, 2004.

[20] P. Felzenszwalb and D. Huttenlocher. Distance transforms of sampled functions. Technical report, Cornell University, 2004.

[21] S. F. Frisken, R. N. Perry, A. P. Rockwood, and T. R. Jones. Adaptively sampled distance fields: a general representation of shape for computer graphics. In *ACM SIGGRAPH Computer Graphics*, pages 249–254, 2000.

[22] C. Galbraith, P. Macmurchy, and B. Wyvill. Blobtree trees. In *Proceedings of Computer Graphics International*, pages 78 –85, 2004.

[23] O. Gourmel, L. Barthe, B. Wyvill, A. Bernhardt, M. Paulin, and H. Grasberger. A gradient-based implicit blend. *ACM Transactions on Graphics*, 32(2):12:1–12:12, 2013.

[24] S. Grosskopf, C. Biermann, K. Deng, and Y. Chen. Accurate, fast, and robust vessel contour segmentation of CTA using an adaptive self-learning edge model. In *Proceedings of SPIE Medical Imaging*, volume 7259, 2009.

[25] A. Guy and B. Wyvill. Controlled blending for implicit surfaces using a graph. In *Proc. of Implicit Surfaces 1995*, pages 107–112, 1995.

[26] H. K. Hahn, B. Preim, D. Selle, and H. O. Peitgen. Visualization and interaction techniques for the exploration of vascular structures. In *IEEE Visualization*, pages 395–402, 2001.

[27] Q. Hong, Q. Li, and J. Tian. Implicit reconstruction of vasculatures using bivariate piecewise algebraic splines. *IEEE Transactions on Medical Imaging*, 31(3):543–553, 2012.

[28] M. A. Jenkins. Algorithm 493: Zeros of a real polynomial [c2]. *ACM TOMS*, 1(2):178–189, 1975.

[29] Y. J. Kim, G. Varadhan, M. C. Lin, and D. Manocha. Fast swept volume approximation of complex polyhedral models. In *Proceedings of the eighth ACM symposium on Solid modeling and applications*, pages 11–22, 2003.

[30] J. Kretschmer, T. Beck, C. Tietjen, B. Preim, and M. Stamminger. Reliable adaptive modelling of vascular structures with non-circular cross-sections. *Computer Graphics Forum (EuroVis)*, 31 (3):1055–1064, 2012.

[31] B. Li, X. Li, K. Wang, and H. Qin. Generalized polycube trivariate splines. In *IEEE Shape Modeling International*, pages 261–265, 2010.

[32] Q. Li. Smooth piecewise polynomial blending operations for implicit shapes. In *Computer Graphics Forum*, volume 26, pages 157–171, 2007.

[33] Q. Li and J. Tian. 2d piecewise algebraic splines for implicit modeling. *ACM Trans. Graph.*, 28(2):13:1–13:19, 2009.

[34] MeVis Medical Solutions AG. MeVisLab: Medical Image Processing and Visualization, http://www.mevislab.de.

[35] T. Moench, M. Neugebauer, and B. Preim. Optimization of Vascular Surface Models for Computational Fluid Dynamics and Rapid Prototyping. In *Second International Workshop on Digital Engineering*, pages 16–23, 2011.

[36] S. Oeltze and B. Preim. Visualization of vasculature with convolution surfaces: method, validation and evaluation. *IEEE Transactions on Medical Imaging*, 24(4):540–548, 2005.

[37] F. H. Post and F. Klok. Deformations of sweep objects in solid modelling. In *Eurographics Conference Proceedings*, pages 103–114, 1986.

[38] A. G. Requicha. Representations for rigid solids: Theory, methods, and systems. *ACM Computing Surveys*, 12(4):437–464, 1980.

[39] A. Ricci. A constructive geometry for computer graphics. *The Computer Journal*, 16(2):157–160, 1973.

[40] R. Schmidt and B. Wyvill. Generalized sweep templates for implicit modeling. In *Proceedings of the 3rd international conference on Computer graphics and interactive techniques in Australasia and South East Asia*, GRAPHITE, pages 187–196, 2005.

[41] W. J. Schroeder, W. E. Lorensen, and S. Linthicum. Implicit modeling of swept surfaces and volumes. In *IEEE Visualization*, pages 40–45, 1994.

[42] C. Schumann, S. Oeltze, R. Bade, B. Preim, and H. Peitgen. Model-free surface visualization of vascular trees. In *IEEE Eurographics Symposium on Visualization*, pages 283–290, 2007.

[43] G. Turk and J. F. O'Brien. Modelling with implicit surfaces that interpolate. *ACM Transactions on Graphics*, 21(4):873, 2002.

[44] J. Tyrrell, E. di Tomaso, D. Fuja, R. Tong, K. Kozak, R. Jain, and B. Roysam. Robust 3-d modeling of vasculature imagery using superellipsoids. *IEEE Transactions on Medical Imaging*, 26(2):223–237, 2007.

[45] J. Van Wijk. Ray tracing objects defined by sweeping planar cubic splines. *ACM Transactions on Graphics*, 3(3):223–237, 1984.

[46] J. Van Wijk. Ray tracing objects defined by sweeping a sphere. *Computers & Graphics*, 9(3):283–290, 1985.

[47] C. Wang and Ö. Smedby. Integrating automatic and interactive methods for coronary artery segmentation: let the PACS workstation think ahead. *International journal of computer assisted radiology and surgery*, 5(3):275–285, 2010.

[48] W. Wang and K. Wang. Geometric modeling for swept volume of moving solids. *IEEE Computer Graphics and Applications*, 6(12):8–17, 1986.

[49] O. Wink, W. Niessen, and M. Viergever. Fast delineation and visualization of vessels in 3d angiographic images. *IEEE Transactions on Medical Imaging*, 19(4):337–346, 2000.

[50] J. Wu, M. Wei, Y. Li, X. Ma, F. Jia, and Q. Hu. Scale-adaptive surface modeling of vascular structures. *BioMedical Engineering OnLine*, 9(1):75, 2010.

[51] B. Wyvill, A. Guy, and E. Galin. Extending the CSG tree. warping, blending and boolean operations in an implicit surface modeling system. In *Computer Graphics Forum*, volume 18, pages 149–158, 1999.

[52] Y. Zhang, Y. Bazilevs, S. Goswami, C. L. Bajaj, and T. J. Hughes. Patient-specific vascular NURBS modeling for isogeometric analysis of blood flow. *Computer methods in applied mechanics and engineering*, 196(29):2943–2959, 2007.