Eurographics Workshop on Visual Computing for Biology and Medicine (2012)
T. Ropinski, A. Ynnerman, C. Botha, and J. B. T. M. Roerdink (Editors)

Visually Guided Mesh Smoothing for Medical Applications

Tobias Moench[1], Christoph Kubisch[2], Kai Lawonn[1], Ruediger Westermann[3], and Bernhard Preim[1]

[1]Department of Simulation and Graphics, University of Magdeburg, Germany
[2]NVIDIA GmbH, Wuerselen, Germany
[3]Computer Science Department, Technische Universitaet Muenchen, Germany

Abstract

Surface models derived from medical image data often exhibit artifacts, such as noise and staircases, which can be reduced by applying mesh smoothing filters. Usually, an iterative adaption of smoothing parameters to the specific data and continuous re-evaluation of accuracy and curvature is required. Depending on the number of vertices and the filter algorithm, computation time may vary strongly and interfere with an interactive mesh generation procedure. In this paper, we present an approach to improve the handling of mesh smoothing filters. Based on a GPU mesh smoothing implementation, model quality is evaluated in real-time and provided to the user as quality graphs to support the mental optimization of input parameters. Moreover, this framework is used to find optimal smoothing parameters automatically and to provide data-specific parameter suggestions.

Categories and Subject Descriptors (according to ACM CCS): Computer Graphics [I.3.5]: Computational Geometry and Object Modeling—Curve, surface, solid, and object representations

1. Introduction

Model generation from tomographic medical image data, e.g., computed tomography (CT) or magnetic resonance imaging (MRI), often requires segmentation of the target structures suffering from image noise and intensity inhomogeneities. These problems are pronounced in medical imaging, where radiation exposure and examination times need to be reduced. The delineation of a target structure often yields binary image data causing strong aliasing artifacts (e.g., "staircases", "terraces") instead of smooth anatomical surfaces. Such discontinuities draw the viewer's attention and distract from relevant features. Combining the segmentation information with the original intensity, e.g., to remove background structures with similar intensity values, allows to restrict surface extraction to the target structure, but may result in local staircase artifacts.

To reduce surface discontinuities, various mesh smoothing filters can be employed as part of a complex model generation pipeline. Anatomical surface models used for surgery or radiation treatment planning put high demands on accuracy, e.g., for estimating distances to neighboring structures or for blood flow simulation in vascular structures. The minimization of artifacts with concurrent preservation of the individual shape and volume requires careful testing of different smoothing methods and parameters. After each parameter change, the resulting model needs to be evaluated w.r.t. curvature reduction and accuracy. Even small delays may disturb during such iterative testing if they appear repeatedly.

Surface model generation is performed for different use cases with specific requirements: within web-based, medical e-learning tools, in computational fluid dynamics (CFD) for blood flow simulation, or in software used by medical experts, such as ENT surgeons or radiology technicians. For example, in radiation treatment planning usually two radiologists perform therapy planning separately based on 3d models. In surgery and treatment planning, high accuracy is required to ensure that, e.g., tumor tissue is completely removed or that a radiation dose is correctly adjusted. In CFD, the loss of features on the aneurysm or parent vessel surface affects flow properties (e.g., wall-shear-stress, flow velocity). Thus, a fast, flexible, and user-friendly integration of smoothing filters into such pipelines is desirable.

In this paper, we present an OpenGL-based framework for interactive real-time mesh smoothing. We control relevant smoothing parameters via mouse-movements similar to brightness and contrast adjustments on radiology workstations and provide immediate visual feedback on accuracy (e.g., volume, local errors) and smoothness. Following this concept, a fast evaluation of model quality allows for a data-specific *preview* of the accuracy and smoothness evolution

DOI: 10.2312/VCBM/VCBM12/091-098

if parameters are modified. We introduce the *model quality graph* as an effective means to perform parameter adjustments in mesh processing. This allows for examining parameter sensitivity and data-specific smoothing suggestions.

2. Related Work

Mesh smoothing filters are applied as part of the model generation pipeline, e.g., to reduce noise within surfaces from laser scanning data or staircase artifacts in models from tomographic medical image data. For binary segmented data, constrained elastic surface nets yield smooth and accurate surfaces [Gib98]. Mesh smoothing filters may cause strong shape and volume changes. Such changes affect the measurement and interpretation of 3d structures in surgical planning, but also the results of blood simulations, e.g., performed for stent implantation planning [MGJ*11]. Depending on the data and application, a large variety of filters is available (e.g., [DMSB99, JDD03, KBSS01, LMJZ09, NISA06, TW03]). Especially the Laplace+*HC* [VMM99] and LowPass [Tau95] filter are suitable for artifact reduction in surfaces from medical image data [BHP06]. Hence, we consider these filters exemplarily, but a generalization of our methods to other data domains and filters is possible. Both filters operate on the topological neighborhood of each vertex and are controlled by the number of iterations (#Its.) and vertex displacement weights. These weights have different meanings which complicates testing and parameter adjustment. For these exemplary filters, computation takes roughly a few seconds for typical anatomical surface models (e.g., $<$500,000 vertices) on a current CPU using only one core. This is, for a pipeline with fixed methods and parameters, not critical. For repeating smoothing tasks in combination with an evaluation of accuracy and artifact reduction, real-time capabilities and a dynamic interface are preferable.

Bade et al. [BHP06] performed a sophisticated analysis on mesh smoothing filters for anatomical surface models to derive parameter suggestions for different shape categories (e.g., compact vs. elongated). Within this study, numerous time-consuming measurements have been performed to find acceptable parameter sets regarding accuracy and smoothness. This correlation between the input parameters and resulting quality measures (curvature, local error, volume) is usually considered as *sensitivity*. Marks et al. [MAB*97] described the idea of "design galleries", which provides precomputed visual results to the user. Berger et al. [BPFG11] presented an interactive approach for the investigation of system dynamics by analyzing parameter sensitivity. Chan et al. [CCM10] determine sensitivity as partial derivative of one variable w.r.t. another, approximated using linear regression. The sensitivity is then used to augment scatterplots.

An example for automated parameter optimization has been presented by Selle et al. [SPSP02]. They estimated optimal thresholds for vessel segmentation and conveyed its effect on the structure's volume. Similarly, Torsney-Weir et al. [TWSM*11] suggested a complex system to replace tedious manual parameter testing by an approach that samples the parameter space and evaluates the results.

Due to the different meanings of smoothing parameters and methods, an estimation of their sensitivity would be helpful and support users during parameter testing and evaluation.

3. OpenGL-Based Mesh Smoothing

For GPU computing, several frameworks, such as CUDA, OpenCL, and DirectCompute are available. The shader concept of OpenGL is ideally suited for mesh operations being closely related to graphical operations, such as displacement mapping or computation of vertex attributes (normals, colors, illumination). Note, that the other frameworks could also be used for computation and their buffers shared with OpenGL for rendering. A pure OpenGL implementation prevents, however, further dependence hardware and SDKs.

3.1. Workflow

OpenGL vertex shaders do natively not provide topological information, such as the location of neighboring vertices. We provide such topology information within the OpenGL shader framework via vertex attributes (vertex buffer objects, VBOs) and large 1d-arrays (texture buffer objects, TBOs) (see Fig. 1b). Within this data structure, each vertex (see Fig. 1a) can access the indices of its 1-ring neighbors via a lookup table (see Fig. 1c) providing information on the number of neighbors and the location (offset) of the indices of neighboring vertices in a neighbor index list (see Fig. 1d). The provided data structure provides fast access to the coordinates of vertex neighbors. A dynamic modification of these arrays, e.g. for topological changes, is not intended.

To enable neighborhood information within vertex shaders, we bind the vertex coordinates buffer, and the buffer with the indices of neighboring vertices as sampler buffers (TBO). For storing the smoothed vertices, we employ a transform feedback buffer (XBO) with the same size as the VBOs (see Fig. 2). The VBOs provide local smoothing weights, e.g. to involve vertex position constraints (e.g., [Gib98]) or

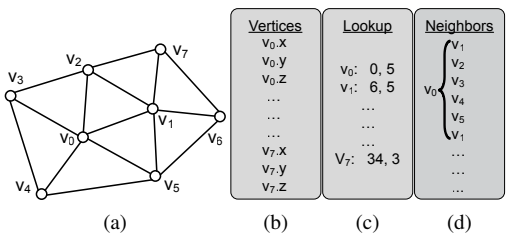


Figure 1: The data is organized in three arrays: vertex coordinates (b), a lookup table (c) holding information on the number of neighboring vertices and their location in the list of neighboring vertices (d).

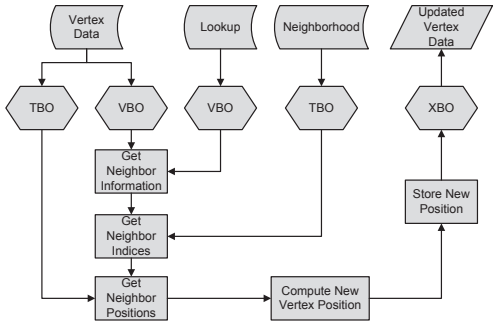


Figure 2: Each vertex catches its neighborhood information and indices of neighboring vertices from differently bound buffers (VBO, TBO) and stores the smoothed position within a transform feedback buffer (XBO).

Model	Vertices	#Neighbors		
		Min	Max	Avg
carotid artery	6,988	4	11	5
sternocleidomastoid muscle (SCM)	38,069	4	15	5
liver	247,772	4	13	5
cerebral aneurysm	6,773	3	12	5

Table 1: The medical surface models employed for evaluation of real-time smoothing capabilities.

to restrict smoothing to artifact areas only [MGJ*11]. After catching neighborhood information from VBOs and TBOs, computing new vertex positions and writing to XBOs, these buffers are switched, such that the modified vertex positions become the input data. Finally, the same workflow is employed to update the vertex normals. With respect to the memory accesses, the lookup of neighborhood information is expensive, since the data might be widely distributed in memory. However, the memory accesses via VBOs and data output via XBO are coalesce.

In this paper, we process surface models from (segmented) tomographic image data with up to 250,000 vertices (e.g., the liver model in Tab. 1). Using an NVIDIA GTX460 as common reference graphics card, frame rates of ≈25 fps are achieved for this comparably large model with 20 iterations of the Laplace+*HC* algorithm. The subsequent normal update is similar to one smoothing iteration with a uniform Laplace filter, which takes less than 1 ms. Similarly, the smaller models yield clearly higher frame rates. Thus, we obtain a completely smooth rendering, even if we would execute mesh smoothing for each render frame.

3.2. Interactive Adjustment of Parameters

The achieved performance gives the opportunity to combine smoothing filters and rendering in an interactive manner. In the reading of radiological image data with a radiology workstation, brightness and contrast adjustments are performed by mouse movements (i.e., horizontal – contrast; vertical – brightness). This feature is used extensively to investigate suspicious regions and variations carefully while the radiologist remains focused on the image data and its features. An interface with separated controls for the viewing parameters would require to interrupt the visual examination of the radiological images.

Borrowing this idea, we can map, e.g., mouse movement direction and distance onto relevant smoothing parameters. Similar to the radiology scenario, this interaction is activated by holding the right or middle mouse button. Any interaction causes an immediate execution of the selected smoothing filter. Thus, the user can remain focused on its visual evaluation of artifact reduction and accuracy preservation. Since users familiar with radiology workstations and exploration of medical image data are among our target users, we can expect a high acceptance of this feature. For occasional users, this should become intuitive very fast.

4. Visual Feedback

Real-time mesh smoothing, combined with an interactive approach, gives an expressive visual feedback and allows for a direct perception of modifications caused by different filters and parameters. The user does, however, not obtain detailed information on the resulting accuracy. Local distance changes (minimum Euclidean distances) and changes of the structure's volume are typical accuracy measures. To determine the influence of mesh smoothing on accuracy, a comparison of the modified mesh M' versus a reference mesh M, regarded as accurate, needs to be performed. For a correct comparison, M should have a high resolution and depict the artifact-free, real shape of the considered structure. For anatomical surface models, we do not have access to such a perfect reference mesh. Hence, we employ the changes between initial mesh M and M' as accuracy indicator.

4.1. Approximation of the Local Error

Within the presented shader-based framework, a comparison of each vertex $v' \in M'$ to each vertex of $v \in M$ would be highly inefficient. Employing the Euclidean distance of v' to its initial position v would roughly reflect the local changes, but does not account for possible drifts along the surface. Thus, we divide error approximation in two steps:

1. the determination of the reference neighborhoods $\mathcal{N}_{ref}$ during mesh smoothing (see Fig. 3a) and
2. the final computation of the minimum Euclidean distance (see Figs. 3b and 3c).

The initial reference neighborhood $\mathcal{N}_{ref}$ of v' is the 1-ring of v. As the position of v' is modified during smoothing, it may tend to drift along the surface (see Fig. 3a). As a consequence, one of the 1-ring neighbors of v might now be closer to v' than v. Thus, this neighbor vertex becomes the new center v_{ref} of $\mathcal{N}_{ref}$. This procedure is repeated after

each smoothing step and moves $\mathcal{N}_{ref}$ according to the drift of v'. After completing the smoothing procedure, we execute the final error computation step. We evaluate all triangles in $\mathcal{N}_{ref}$ by testing whether v' is located inside the prism spanned by each of the triangles (see Fig. 3b) and the corresponding normal. This can be solved by:

$$(a,b,c) = P^{-1}(v' - v_{ref}). \quad (1)$$

The columns of the matrix P are the vectors e_1, e_2 along the triangle edges incident to v_{ref}, and the triangle normal n_i. P^{-1} exists only for non-degenerated triangles. If $a,b \geq 0$ and $a+b \leq 1$, v' is inside the prism of the considered triangle. Moreover, if n_i is orthonormal to e_1 and e_2, c returns the distance from v' to the triangle plane ($d = c$). In case v' is not inside any of the prisms of $\mathcal{N}_{ref}$, v' might be close to one of the vertices or edges of $\mathcal{N}_{ref}$ (see Fig. 3c). As a special case, v' might have left $\mathcal{N}_{ref}$, but the closest vertex is still v_{ref}. In those cases, we determine d as the minimum Euclidean distance towards the edges of $\mathcal{N}_{ref}$.
By moving $\mathcal{N}_{ref}$ we may get stuck locally, and thus, not find the global minimum Euclidean distance to M. This occurs, if, e.g., a smoothed vertex is getting close to a topologically very distant part of the model (e.g., another branch of a vascular structure). This real minimum is, however, not important to the error feedback.

4.2. Curvature

A decrease of curvature is an indicator for the degree of smoothing. Thus, within the same final computation step executed for error computation (see Sec. 4.1), we estimate the curvature. Within the vertex shader, we determine the discrete Gaussian curvature K_G at a vertex v_i by iterating over all adjacent triangles ($\#f_i$) considering the area A_{Mixed} and the angle θ_j between the edges incident to v_j of each triangle [MDSB03]:

$$K_G(v_i) = (2\pi - \sum_{j=1}^{\#f_i} \theta_j)/A_{Mixed}. \quad (2)$$

5. Sensitivity Analysis

The immediate rendering update in combination with curvature or error coloring provides a fast overview on the local effects. By modifying relevant parameters continuously via mouse movement (see Sec. 3.2) and observing the changes, the user performs a mental sensitivity analysis to explore the correlation between input parameters and smoothing results. Since parameters of different filters may have different meanings, an intuitive usage is not guaranteed. For example, the weights of the Laplace+*HC* method describe the influence of the original vertex positions. Thus, higher values represent lesser smoothing. In this section, we describe how we support this analysis visually and quantitatively.

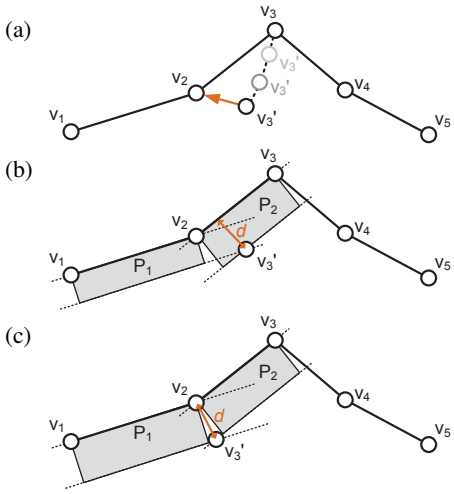


Figure 3: 2d illustration of the error approximation. (a) $\mathcal{N}_{ref}$ consists of v_2, v_3, v_4. After some iterations, v_2 is the closest vertex and $\mathcal{N}_{ref}$ comprises v_1, v_2, v_3. (b) P_1, P_2 represent the prisms spanned by v_1, v_2, v_3. (c) The prism test fails, if v_3' is not inside of P_1 or P_2.

5.1. Volume Computation

Besides local distances, volume preservation is a major criterion for the accuracy of mesh smoothing. For a closed surface, the computation of the volume of a model may be determined from the discrete form of the divergence theorem [DMSB99] (see Eq. 3). Following this equation, we determine the volume by iterating over all m triangles and employ the face normal $\hat{n}_i$, the area A_i, and an arbitrary point v_i on each triangle. Within our framework, this can be achieved efficiently by employing geometry shaders, which allow for computations per primitive. The volume fractions of the triangles are accumulated via parallel prefix sums [Ble93].

$$V = \frac{1}{3} \sum_{i=0}^{m} v_i \cdot \hat{n}_i A_i \quad (3)$$

5.2. Model Quality Graphs

To further improve the handling of mesh smoothing filters, we introduce *model quality graphs*. These graphs display the correlation between a smoothing parameter and model quality. The current setting is visually indicated by a vertical line. After initially loading the model and building the data structures, we execute an initial computation of the quality metrics. For that, we increase each smoothing parameter within its defined bounds (e.g., weights $\in [0,1]$ with step size 0.05). During modification of one parameter, all others remain constant. After each smoothing operation, the mesh is again evaluated w.r.t. local distance changes, volume shrinkage, and curvature reduction. This procedure is repeated for

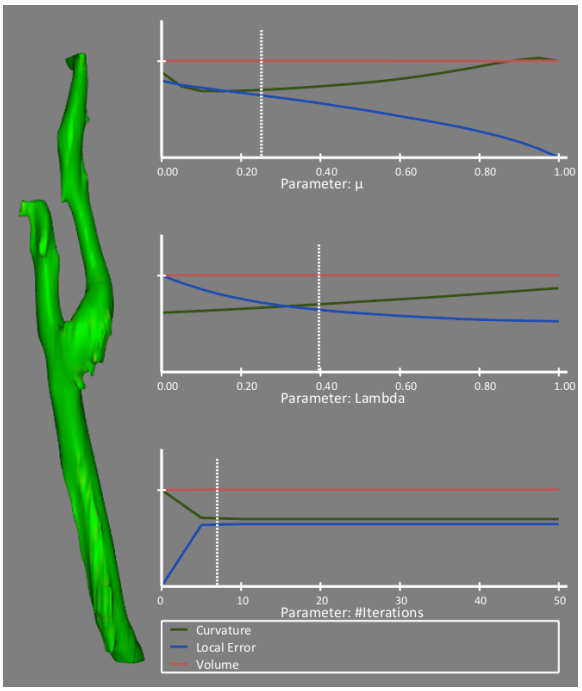


Figure 4: Prediction of model quality for the carotid artery, color-coded by the local error (Laplace+*HC* with 3 parameters – #Its. & smoothing weights λ, μ).

all desired input parameters. Figure 4 shows model quality graphs for a surface model of the carotid artery. In this example, three input parameters of the Laplace+*HC* filter have been considered. In contrast to standard Laplacian smoothing, two parameters (λ, μ) control the influence of original and intermediate vertex positions. Thus, stronger smoothing is achieved for small values of λ and μ. Without the model quality graphs, this might be confusing to occasional users.
This initial guess on the evolution of model quality is available to the user before any manual parameter adjustments. The quality graphs are updated after each interaction step. Scaling of the y-axis is performed depending on the quality measures of the initial surface model without smoothing. For each input parameter (and diagram), a value hint is displayed (dashed vertical lines in Fig. 4), which helps the user to visually correlate the current parameter set with the precomputed quality measures. The user can also modify each smoothing parameter via the graphs. For integer parameters the value hint is aligned accordingly.
The described concept of quality graph previewing supports the user during the interactive parameter optimization and shall decrease the amount of trial and error cycles during parameter testing and model inspection. The graph-based quality preview provides a direct feedback on the influence of each input parameter. Thus, it assists the mental sensitivity analysis performed by the user. This feedback supports primarily experienced users and enables them to adjust the parameters efficiently. Occasional users will still need a few trial and error cycles. The graphs enhance the visual feedback provided by the (curvature-/error-)colored, smooth mesh with quantitative measures.

5.3. Parameter Suggestions

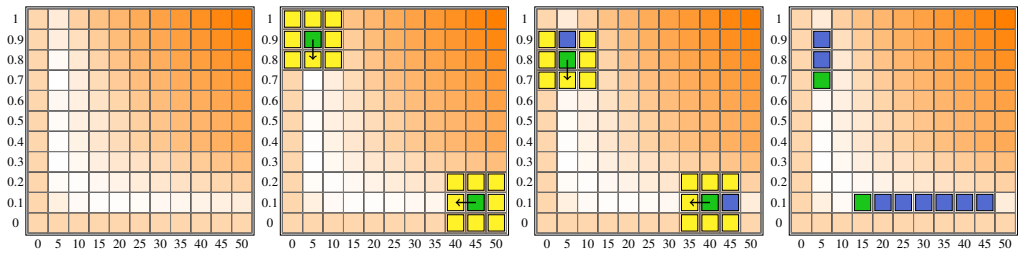
By analyzing the quality measures automatically, we generate parameter suggestions to further simplify manual testing. This is used for an initial suggestion of model-specific default values. Thus, a large number of parameter combinations needs to be evaluated w.r.t. accuracy and smoothness. For an initial guess, a rough parameter sampling can be chosen, e.g. a step size of 5 or 10 for #Its. $\in [0,50]$, or a step size of 0.1 for weighting factors $\in [0,1]$. By executing smoothing and mesh evaluation for all parameter combinations, we compute the quality score S for each parameter set separately as a weighted sum of the total curvature K, the average local error E, the volume V, the initial curvature and volume K_0, V_0, and the target smoothness coefficient τ:

$$S = \tau \frac{K}{K_0} + (1-\tau)\frac{1}{2}\left(\frac{E}{E+1} + \frac{|V-V_0|}{V_0}\right). \qquad (4)$$

The weighted sum is controlled by the target smoothness $\tau \in [0,1]$ describing the tradeoff between accuracy and smoothness. τ maps all n input parameters onto a single one. Hence, it may be used to bypass interaction with several separated parameters. The optimal parameter set is then found by minimizing S. For $\tau \to 1$, the suggestion considers more smoothness, whereas it yields higher accuracy for $\tau \to 0$. By default, τ is set to 0.5. The effect of τ can be seen in Tab. 2. For the carotid artery (see Fig. 4), the average curvature increases with stronger smoothing, since thin parts tend to collapse. Our method is sensitive to such changes. The liver model is more compact and has a large volume. Smoothing can thus be stronger compared to elongated structures.
This optimization yielded stable and plausible results for all employed models and took up to 400 ms for small and medium sized models (e.g., SCM and carotid artery in Tab. 1). For large models, such as the liver, computation takes about 6 s, which is still acceptable, since the optimization is

τ	Carotid artery		SCM		Liver	
	λ	#Its.	λ	#Its.	λ	#Its.
0.0	0.0	0	0.0	0	0.0	0
0.2	0.3	5	0.6	5	0.7	10
0.4	0.6	5	0.7	10	0.8	30
0.6	0.9	5	0.8	15	0.8	45
0.8	1.0	5	0.9	50	1.0	50
1.0	1.0	15	1.0	50	1.0	50

Table 2: Parameter suggestions for modifying τ for differently shaped structures (uniform Laplace, 2 parameters).



(a) Sample parameter space (b) Initial local minimization (c) Neighborhood search (d) Minimization finished

Figure 5: Examples for iterative score optimization for two parameters (λ, #Its.) (a). Beginning from two positions (b) in parameter space, similar parameter sets are evaluated. (c) We proceed with the parameter set, which yielded the minimum scores in each neighborhood until a local minimum is found (d).

performed only once during model initialization. For more input parameters, computation times increase accordingly.
As a remedy, we suggest an iterative, local search in the spanned parameter space of S (see Fig. 5a). The precomputation of K, E, V, and S is not required. Starting at a given position in the parameter space, we perform a local gradient search. We consider all parameter sets defined by the direct $3^n - 1$-neighborhood and perform smoothing, evaluation, and score computation (see Fig. 5b for the 2d case). The parameter set which yielded the minimum score, serves as next search center. We proceed until a local minimum has been found (see Figs. 5c and 5d). Depending on the number of input parameters n, we create n search patterns. For example, for two input parameters (#Its., λ), two search patterns are initialized. Each of these patterns begins at the maximum value of one parameter, e.g., at $\lambda = 1.0$, #Its.$= 0$ and $\lambda = 0.0$, #Its.$= 50$. This yields at most n local minima, from which we select the lowest value. Since the quality scores for the complete parameter space are not known, we cannot guarantee to find the global minimum, but the use of n search patterns, starting from opposite sides of the parameter space, yielded the global minimum for most test cases. Otherwise, the suggested parameters were always valid alternatives, since they were located in an area with similar quality scores as the global minimum. The iterative optimization was up to $6\times$ faster than an evaluation of the complete parameter space. The use of only one search pattern is faster, but increases the likelihood that the optimum is not found.
During user interaction, the suggestions for the other parameters are updated. Each parameter is again modified separately, while all others remain constant. For each modified parameter we obtain curves for the quality measures and scores according to Eq. 4. The location of the minimum score yields the suggestion for the specific parameter.

6. Application

The described methods are the core of a real-time smoothing framework of medical surface meshes. Within *fixed* model generation pipelines, our framework is employed to select smoothing parameters automatically based on a predefined target smoothness τ. During interaction, model quality graphs and τ guide the user.

6.1. Interactive Model Generation

Our framework is utilized within a prototype software for the segmentation and surface extraction of structures for building physical models using rapid prototyping technology. The resulting phantoms are utilized for training of interventional techniques, procedures and teaching. Thus, surface and physical models are required to depict the individual anatomical shape and features to provide a realistic look and resemble the intraoperative situation. Within this context, medical experts operate the software which comprises segmentation and model generation. Interaction with the software has to be intuitive, fast and avoid complex parameter tuning.
Mesh smoothing is applied as subsequent to a real-time Marching Cubes implementation (similar to [DZTS08]), which yields an immediate surface update after isovalue adjustment. Models range between 50k and 300k vertices. With the immediate (graph-based) quality feedback, smoothing does not interfere with this highly interactive procedure. The mapping of smoothing parameters to mouse movements integrates well to the interaction schemes known by medical experts. Our reliable, initial parameter suggestions, utilized to minimize iterative parameter testing.

6.2. Web-based Medical e-Learning

Web-based rendering (e.g., via X3D, WebGL) in combination with higher bandwiths of current internet connections becomes increasingly important. In medicine, web-based e-learning platforms arise to share and discuss clinical cases. This comprises tomographic data, but also explorable surface renderings. Such models may be provided by platform users or dedicated authors that are responsible. In both cases, it is most likely that several models need to be generated, e.g., to describe spatial relations of separated structures.

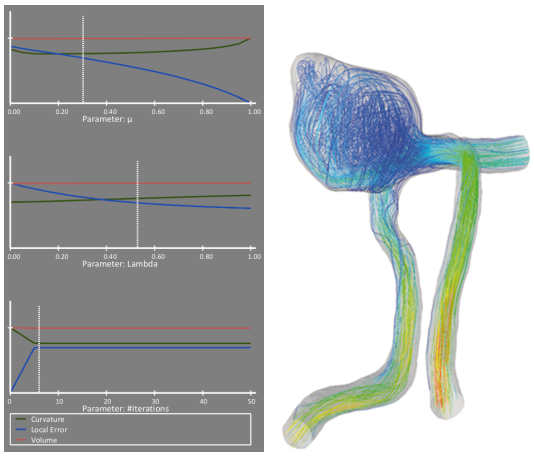
We have integrated our methods into the model generation pipeline of a web-based e-learning platform that incorporates Web 2.0 functionality to enhance the cooperation between surgeons. The users providing content for this e-learning platform prepare 3D models according to the requirements of a medical expert, e.g., a liver surgeon. These users, e.g., technical assistants or co-workers in medical informatics, are familiar with model generation. Since usually several models need to be prepared, manual effort should be reduced, but still allow for full control. This functionality is provided by our model quality graphs and data-specific parameter suggestions.

6.3. Simulation of Blood Flow

CFD of vascular systems is performed to estimate the rupturing risk of aneurysms and the ability to reduce this risk, e.g., by inserting stents. Accurate and smooth surface models are used as input for the patient-specific simulation of blood flow [CCA*05, CLS*01] (see Fig. 6). Fine details on the surface influence the granularity of the resulting volume mesh and finally the quality of CFD results (e.g., wall-shear-stress, flow velocity) [MGJ*11]. The extraction of a surface from the image data involves a lot of manual effort to remove artifacts caused by, e.g., low resolution, partial volume, beam hardening effects, or insufficient contrast agent distribution [MNP11]. This procedure is usually executed by people with long experiences in segmentation and model generation. Permanent feedback of medical experts is required to differentiate between surface artifacts and pathological deviations. CFD simulation raises several constraints, which also need to be considered during model generation. Thus, the pipeline is often executed several times for the same data. To reduce this effort, the methods within each step should be easy to configure and should not introduce disturbing waiting times. Our framework integrates these characteristics, since resulting errors can be evaluated immediately and parameters can easily be adjusted according to different requirements. By employing the quality score, we can obtain appropriate smoothing suggestions automatically without the need for manually adjusting the smoothing parameters after the model input has been changed in earlier pipeline steps.

6.4. Evaluation

We performed a small informal evaluation with four participants to test mouse movement smoothing, interaction with the quality graphs, and plausibility of smoothing suggestions. All volunteers had a strong background in medical visualization, only two had sophisticated knowledge in model generation, and one of these was familiar with mesh smoothing algorithms. All participants confirmed that model quality graphs in combination with the interaction concept are helpful and intuitive. After a brief introduction and getting familiar with the interface its interaction options (mouse



(a) (b)

Figure 6: (a) Model quality graphs for a cerebral aneurysm (Laplace+*HC* filter). (b) The aneurysm model with embedded flow information obtained from CFD simulation.

movement, parameter selection in the graphs), they began to explore the effect of each parameter. Afterwards they were told to smooth two differently complex models (carotid artery, petrous bone) considering accuracy. All participants started rotating the models to explore it and to look for artifacts. Subsequently they concentrated on the quality graphs and adjusted the parameters intuitively to a parameter set yielding an initial strong curvature reduction. Thereby, we observed two issues: an initial exploration of the parameter influences and rough adjustments were performed using mouse movements to observe the model behavior and quality graphs. Precise parameter tuning was often accomplished by selection in the graphs. The adjustments in the quality graphs were more careful than during model observation.

The interaction with two parameters of uniform Laplace was trivial, since both parameters (#Its., λ) yield stronger smoothing for higher parameters. For the Laplace+*HC* filter, three volunteers were confused after the first look at the quality graphs. However, even without algorithm knowledge they could easily adjust smoothing parameters to their needs. The parameter suggestions were considered as plausible, but were more careful, than the user-specified parameter sets. The initial suggestions were often slightly modified to evaluate small parameter changes. Two participants mentioned, that they prefer the initial suggestions when preparing several models within a complex pipeline and use the graphs as visual validation of the automated smoothing result.

All participants remarked that the labeling of the y-axis was not completely clear, since three different curves were presented within each graph. A uniform labeling and scaling is, however, not trivial. The scaling of the error graphs was slightly misleading, since they were scaled according to the currently occurring maximum error. Thus, the absolute er-

ror is not clear. This might be resolved by scaling with the specific voxel diagonal. Moreover, horizontal reference lines could help provide a better feedback on the strength of accuracy loss or curvature reduction.

7. Conclusion

We described how different applications working with anatomical surface models benefit from interactive real-time mesh smoothing. Using OpenGL, mesh smoothing filters have been combined with a mapping of mouse movements to their input parameters. This is strongly motivated by the typical interactions of the intended users (e.g., radiologists, radiology technicians) with medical image data. The interaction scheme is not intended to set parameters precisely, but provides insight to the influence of each parameter. We presented how quality measures (local error, volume, curvature) are determined efficiently to provide visual feedback. The achieved performance enabled *model quality graphs* for visually guiding the user during parameter adjustments. By evaluating accuracy and smoothness for a large variety of parameter sets, model-specific parameter suggestions can be made. The quality graphs and parameter suggestions are especially relevant, since the weighting factors of different smoothing methods may have different meanings.
User guidance in terms of *model quality graphs* and parameter suggestions is crucial for comparing the effects of different parameters. Especially occasional users without a strong background in mesh smoothing methods benefit from the previewing functionality. In the reading of tomographic image data, e.g., radiologists are highly familiar with comparing images arranged in galleries. By providing a gallery of differently smoothed (and color-coded) instances of initial model, the handling of mesh smoothing might be further adapted to medical workflows.

Acknowledgments

Tobias Moench is funded by the German Federal Ministry of Education and Research (BMBF) within the ViERforES-II project (no.01IM10002B).

References

[BHP06] BADE R., HAASE J., PREIM B.: Comparison of fundamental mesh smoothing algorithms for medical surface models. In *Proc. of Simulation und Visualisierung* (2006), pp. 289–304. 2

[Ble93] BLELLOCH G. E.: Prefix sums and their applications. In *Synthesis of Parallel Algorithms*. Morgan Kaufmann, 1993. 4

[BPFG11] BERGER W., PIRINGER H., FILZMOSER P., GRÖLLER E.: Uncertainty-aware exploration of continuous parameter spaces using multivariate prediction. *Computer Graphics Forum 30*, 3 (2011), pp. 911 – 920. 2

[CCA*05] CEBRAL J. R., CASTRO M. A., APPANABOYINA S., PUTMAN C. M., MILLAN D., FRANGI A. F.: Efficient pipeline for image-based patient-specific analysis of cerebral aneurysm hemodynamics: technique and sensitivity. *IEEE Trans. Med. Imaging 24*, 4 (2005), 457–467. 7

[CCM10] CHAN Y.-H., CORREA C., MA K.-L.: Flow-based scatterplots for sensitivity analysis. In *IEEE VAST* (2010), pp. 43 –50. 2

[CLS*01] CEBRAL J. R., LÖHNER R., SOTO O., CHOYKE P. L., YIM P. J.: Patient-specific simulation of carotid artery stenting using computational fluid dynamics. In *Proc. of MICCAI* (2001), pp. 153–160. 7

[DMSB99] DESBRUN M., MEYER M., SCHRÖDER P., BARR A. H.: Implicit fairing of irregular meshes using diffusion and curvature flow. In *Proc. of ACM SIGGRAPH* (1999), pp. 317–324. 2, 4

[DZTS08] DYKEN C., ZIEGLER G., THEOBALT C., SEIDEL H.-P.: High-speed marching cubes using histopyramids. *Computer Graphics Forum 27*, 8 (2008), 2028–2039. 6

[Gib98] GIBSON S. F. F.: Constrained elastic surface nets: Generating smooth surfaces from binary segmented data. In *Proc. of MICCAI* (1998), pp. 888–898. 2

[JDD03] JONES T. R., DURAND F., DESBRUN M.: Non-iterative, feature-preserving mesh smoothing. *ACM Trans. Graph. 22*, 3 (2003), 943–949. 2

[KBSS01] KOBBELT L. P., BOTSCH M., SCHWANECKE U., SEIDEL H.-P.: Feature sensitive surface extraction from volume data. In *Proc. of ACM SIGGRAPH* (2001), pp. 57–66. 2

[LMJZ09] LI Z., MA L., JIN X., ZHENG Z.: A new feature-preserving mesh-smoothing algorithm. *Vis. Comput. 25*, 2 (2009), 139–148. 2

[MAB*97] MARKS J., ANDALMAN B., BEARDSLEY P. A., FREEMAN W., GIBSON S., HODGINS J., KANG T., MIRTICH B., PFISTER H., RUML W., RYALL K., SEIMS J., SHIEBER S.: Design galleries: a general approach to setting parameters for computer graphics and animation. In *Proc. of ACM SIGGRAPH* (1997), pp. 389–400. 2

[MDSB03] MEYER M., DESBRUN M., SCHRÖDER P., BARR A. H.: Discrete differential-geometry operators for triangulated 2-manifolds. In *Visualization and Mathematics III*, Hege H.-C., Polthier K., (Eds.). Springer-Verlag, 2003, pp. 35–57. 4

[MGJ*11] MOENCH T., GASTEIGER R., JANIGA G., THEISEL H., PREIM B.: Context-Aware Mesh Smoothing for Biomedical Applications. *Computers and Graphics 35(4)*, 4 (2011), 755 – 767. 2, 3, 7

[MNP11] MOENCH T., NEUGEBAUER M., PREIM B.: Optimization of Vascular Surface Models for Computational Fluid Dynamics and Rapid Prototyping. In *Proc. of IWDE* (2011), pp. 16–23. 7

[NISA06] NEALEN A., IGARASHI T., SORKINE O., ALEXA M.: Laplacian mesh optimization. In *Proc. of GRAPHITE* (2006), pp. 381–389. 2

[SPSP02] SELLE D., PREIM B., SCHENK A., PEITGEN H.-O.: Analysis of vasculature for liver surgical planning. *IEEE Transactions on Medical Imaging 21*, 11 (2002), 1344 –1357. 2

[Tau95] TAUBIN G.: A signal processing approach to fair surface design. In *Proc. of ACM SIGGRAPH* (1995), pp. 351–358. 2

[TW03] TASDIZEN T., WHITAKER R.: Anisotropic diffusion of surface normals for feature preserving surface reconstruction. In *Proc. of 3-D Imaging and Modeling* (2003), pp. 353–360. 2

[TWSM*11] TORSNEY-WEIR T., SAAD A., MOLLER T., HEGE H., WEBER B., VERBAVATZ J., BERGNER S.: Tuner: Principled parameter finding for image segmentation algorithms using visual response surface exploration. *IEEE TVCG 17*, 12 (2011), 1892 –1901. 2

[VMM99] VOLLMER J., MENCL R., MÜLLER H.: Improved laplacian smoothing of noisy surface meshes. *Computer Graphics Forum 18*, 3 (1999), 131–138. 2