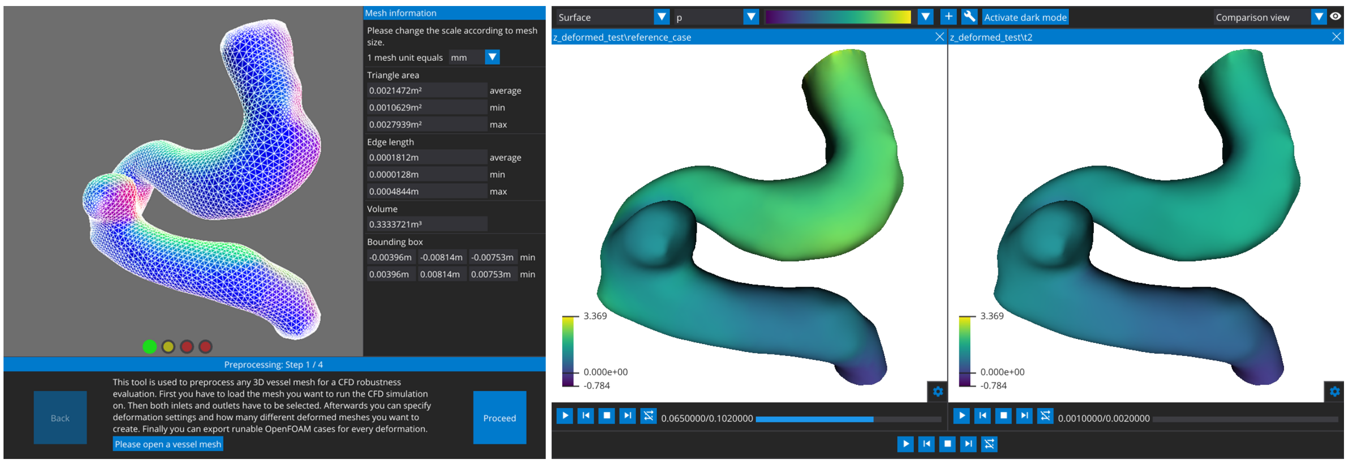
Eurographics Workshop on Visual Computing for Biology and Medicine (2019)
B. Kozlíková, L. Linsen, and P.-P. Vázquez (Editors)

Robustness Evaluation of CFD Simulations to Mesh Deformation

A. Scheid-Rehder[1], K. Lawonn[1], M. Meuschke[2]

[1]Institute of Computational Visualistics, University of Koblenz - Landau, Germany
[2]Department of Simulation and Graphics, University of Magdeburg, Germany


Mesh information

Please change the scale according to mesh size.

1 mesh unit equals mm

Triangle area

0.0021472m²	average
0.0010629m²	min
0.0027939m²	max

Edge length

0.0001812m	average
0.0000128m	min
0.0004844m	max

Volume

0.3333721m³

Bounding box

-0.00396m	-0.00814m	-0.00753m	min
0.00396m	0.00814m	0.00753m	min

Preprocessing: Step 1 / 4

This tool is used to preprocess any 3D vessel mesh for a CFD robustness evaluation. First you have to load the mesh you want to run the CFD simulation on. Then both inlets and outlets have to be selected. Afterwards you can specify deformation settings and how many different deformed meshes you want to create. Finally you can export runable OpenFOAM cases for every deformation.

Please open a vessel mesh

Surface p

z_deformed_test\reference_case

0.0650000/0.1020000

z_deformed_test\t2

0.0010000/0.0020000

Figure 1: *Main layouts of the proposed tool. The first pre-processing step shows some information about the imported mesh (left). The default visualization layout employs a side-by-side view of the two currently loaded cases (right).*

Abstract

CFD simulations are an increasingly important method for the non-invasive analysis of risk factors for aneurysm rupture. Their robustness, however, has to be examined more thoroughly before clinical use is possible. We present a novel framework that enables robustness evaluation of CFD simulation according to mesh deformation on patient-specific blood vessel geometry. Our tool offers a guided workflow to generate, run, and visualize OpenFOAM simulations, which significantly decreases the usual overhead of CFD simulations with OpenFOAM. Besides, the deformation of the original geometry allows the user to evaluate the robustness of the simulation without the need to repeat expensive operations of the data pre-processing phase. We assessed the robustness of CFD simulations by applying our framework to several aneurysm data sets.

CCS Concepts

• ***Human-centered computing*** → ***Scientific visualization;***

1. Introduction

Cerebral aneurysms are pathological vessel wall dilations of intracranial arteries that occur in about 6% of the general population. Over time, the diseased area may growth, causing the vessel wall to become thinner and the aneurysm to rupture [BSN06]. Aneurysm rupture leads to internal bleeding, which is associated with high morbidity and mortality [NSA*09].

Advances in medical imaging and its frequent use in healthcare are leading to increased detection of unruptured aneurysms. There are several ways to treat unruptured aneurysms, each with its advantages and risks [CMG11]. However, operative risks can exceed the risk of spontaneous rupture [Wie03]. Moreover, treatment might not be necessary as 50% to 80% of aneurysms will never rupture [BSN06]. Therefore, it is essential to determine the patient-specific rupture risk and limit treatment to high-risk patients.

Several factors seem to influence aneurysm rupture risk such as genetics, vessel morphology, and hemodynamics, which are not well understood. Morphological features include aneurysm size, location, and shape, while hemodynamic factors comprise quantitative attributes such as wall shear stress (WSS), and qualitative data,

DOI: 10.2312/vcbm.20191244

i.e. flow patterns. Computational fluid dynamics (CFD) simulations enable a non-invasive analysis of hemodynamics. Due to growing advances in computational speed, CFD simulations will become more accessible and accurate, and could, therefore, become essential in clinical practice [WP11].

Patient-specific data are crucial to assess rupture risk. While, the patient-specific vessel geometry can be extracted from medical image data, *in vivo* hemodynamics such as flow rates are rarely available. Thus, researchers have to rely on generalized assumptions, which are a significant source of errors [CC15]. Moreover, during the process from imaging to 3D model reconstruction inaccuracies can be introduced that could significantly alter the outcome of CFD simulation [BSV*18; SQAM14]. To reduce acceptance problems of CFD simulations for aneurysm risk assessment in the clinical environment, further studies on the robustness of these simulations have to be performed [CC15].

We present a novel tool to evaluate the robustness of CFD simulations on patient-specific geometry. It provides a guided workflow with three major components, namely, pre-processing, CFD simulation, and visualization. During pre-processing, the user can create various deformations of the original geometry. Then, an OpenFOAM case with user-specific settings is generated for the initial geometry as well as the deformed cases. We use mesh generation utilities to create volume grids, CFD solvers, and post-processing utilities. The simulation results are comparatively depicted using various visualization techniques, see Figure 1.

2. Related Work

Related work comprises the robustness evaluation of aneurysm blood flow simulations and the visual exploration of the results.

2.1. Robustness of Aneurysm Blood Flow Simulations

Simulated blood flow data is usually computed on patient-specific geometry through CFD software. Berg *et al.* [BJT12] compared the commercial CFD solver *ANSYS Fluent* with the open-source software *OpenFOAM*. While there were some small differences in the results, both solvers were generally in good agreement. In terms of setup, however, Fluent was found to be far more intuitive than OpenFOAM, mainly due to its graphical user interface.

Chung and Cebral [CC15] described CFD as a promising tool for clinical practice. However, more research on its robustness is needed. Several authors conducted studies on the robustness of aneurysm simulations to both model assumptions and geometric factors. Xiang *et al.* [XTK*12] investigated the influence of the assumption of Newtonian blood rheology on scalar values. Newtonian models could overestimate shear rate and WSS in regions of slowly recirculating flow, which might alter the diagnosis. Schneider *et al.* [SMA*13] investigated the influence of conventional angiography (2D DSA) and 3D rotational angiography (RA) on simulation results. Hemodynamics inside the aneurysm might change due to over-estimated neck sizes that resulted from the use of 3D RA, which is considered the reference standard. Both different flow patterns and impingement zones were found, and WSS values were shown to change quite considerably in some cases.

Valen-Sendstad and Steinman [VS14] analyzed the influence of the resolution of the CFD solution strategy on the computed variables. Normal-resolution simulations are feasible for rupture risk prediction as long as comparable studies are available for reference, and highly reduced hemodynamic factors are used. However, high-resolution simulations should be preferred as normal-resolution strategies may mask relevant correlations and regions with fluctuating WSS may not be detected as the lower resolution cannot detect these effects. Valen-Sendstad *et al.* [VBS*18] conducted a challenge on the topic of CFD simulations comprising five aneurysm cases. The teams were free to choose a segmentation method and simulation settings. Results showed that different segmentation methods could produce differences in WSS of up to 65%. Thus, even minor changes in segmentation could have non-negligible effects on hemodynamics. A study by Sen *et al.* [SQAM14] showed that the volume difference between a manual and several automatic segmentation methods averaged 9.3% when segmenting a cerebrovascular vessel structure. Recently, Berg *et al.* [BSV*18] analyzed the effects of the choice of DSA reconstruction kernel on aneurysmal hemodynamics. They found apparent differences in WSS and inflow velocity at the ostium, *i.e.* the aneurysm opening.

2.2. Visualization of Simulation Results

Visualizing flow data is essential to enable the exploration of simulation results. Preim and Botha [PB13] provided a summary of the visualization of blood flow data. The most prominent approaches are volume rendering of scalar values and flow visualization through stream- or path lines. An efficient GPU-based ray-casting method for volume rendering is presented by Krüger *et al.* [KW03] where a proxy geometry of the object bounding box is utilized to generate rays. Mattausch *et al.* [MTHG03] described several strategies, such as halos to improve streamline visualization. Gasteiger *et al.* [GNBP11] presented an interactive lens to facilitate simultaneous exploration of vessel morphology and flow data. Lawonn *et al.* [LGP14] presented a visual reduction of the vessel surface to show internal blood flow where morphological features can still be perceived.

However, these techniques are affected by visual clutter. Clipping planes are helpful to remove occluding streamlines but might remove relevant information. Clustering and classification can be performed to reduce visual complexity. Kuhn *et al.* [KLG*11] presented a visualization approach for vector fields where the domain is clustered into regions based on the bending energy of streamlines. Thereby, they are able to extract regions of potential interest where cluster complexity is high. The clusters are visualized as semi-transparent surfaces and colored depending on the flow behavior. Oeltze *et al.* [OLK*14] compared the results of three state-of-the-art clustering algorithms on streamlines. Furthermore, they displayed a representative of each cluster rather than all streamlines. Recently, Meuschke *et al.* [MOB*18] used the results of the line clustering to assign the flow pattern of each cluster to one of several predefined classes.

CFD data is interpolated onto the vessel surface to display scalar data such as WSS magnitude. To provide a more unobstructed view on the scalar data, Goubergrits *et al.* [GSK*11] projected the aneurysm surface onto a uniform sphere. This allowed

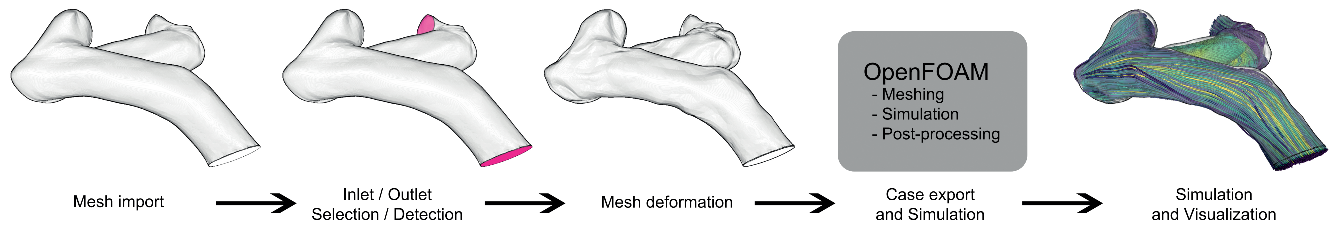

Mesh import
Inlet / Outlet Selection / Detection
Mesh deformation
OpenFOAM
- Meshing
- Simulation
- Post-processing
Case export and Simulation
Simulation and Visualization

Figure 2: *Overview of our guided pipeline that consists of several pre-processing steps, the simulation and the visualization of the results.*

them to create statistical 2D maps which show the WSS distribution in ruptured and unruptured cases, respectively. Meuschke *et al.* [MGB*19] introduced a tool that employs statistical graphics such as scatter plots to explore scalar data. Additionally, they presented two techniques for the simultaneous investigation of scalar data both on the vessel surface and on a 2D map representation.

3. Data Acquisition and Pre-Processing

In this section, we describe the acquisition and pre-processing of the data as input for our framework.

Acquisition. At first, CTA, MRA, and 3D RA image data of the aneurysm morphology are acquired. Common image resolutions are up to $512 \times 512 \times 140$ with a voxel size of $0.35 \times 0.35 \times 0.9$mm.

3D Surface Mesh Extraction. From the clinical image data, the vessel surface is reconstructed using the pipeline by Mönch *et al.* [MNP11]. Due to the used contrast agents, all employed modalities exhibit high vessel-to-tissue contrast. Therefore, a threshold-based segmentation followed by a connected component analysis is used to separate the aneurysm and its parent vessel from the surrounding tissue. Marching Cubes [LC87] is applied to the segmented data to extract the 3D vessel surface. To use the 3D mesh as input for the CFD simulations, it is necessary to manually correct artifacts and to clip the vessel geometry at inlets and outlets [MNP11]. Lastly, the mesh quality is optimized through a combination of metric and topological changes [Sch97].

4. Robustness Analysis Framework

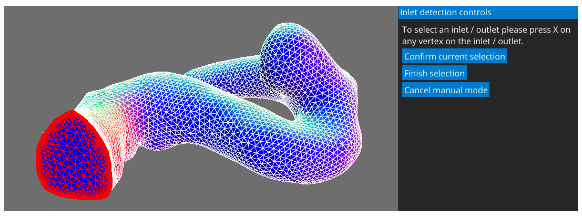
Our tool guides the user through the complete CFD simulation and visualization process. While the tool focuses on robustness analysis of simulation results w.r.t deformations of the input geometry, it can also be used to perform CFD simulations and examine the results easily. Figure 2 shows the pipeline of our tool. During preprocessing, the patient-specific vessel mesh is imported, inlets and outlets are detected, and multiple variations of the original mesh are generated. Afterwards, the user can set several configuration options for the simulation, and the OpenFOAM files for each mesh variation are generated. Then, the OpenFOAM utilities for volume mesh generation and the simulation itself are executed. Finally, the results are visualized using several techniques such as volume rendering and streamlines, both in a single case view and in a comparative view where the original mesh and a deformed mesh are shown side-by-side. In the following, each step is described in more detail.

4.1. Pre-Processing

We would like to provide an easy-to-use and guided OpenFOAM case generator, as the configuration of cases can be difficult and usually requires an enormous amount of manual setup. Especially the lack of a GUI is a considerable disadvantage of OpenFOAM. Mainly, however, meshes with small deviations from an original geometry should be created in order to evaluate the robustness of CFD simulations. These variations mimic differences introduced by 3D surface mesh reconstruction from patient-specific image data [SQAM14]. To evaluate the robustness of simulations, you could change any variable in one of these steps and then run the simulation again to look for differences. This, however, is not feasible, as the entire reconstruction pipeline is time-consuming. Therefore, we modify the acquired triangle mesh and simplify the workflow considerably. The deformation locations are randomized to simulate geometric variation due to, *e.g.* a different segmentation kernel. Nevertheless, the user can adapt the deformation settings.

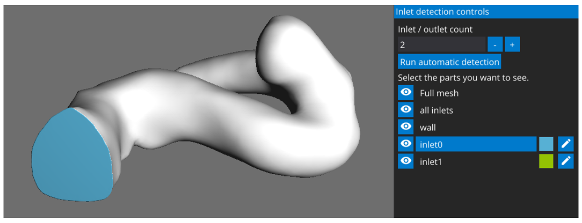
Mesh Import. First, the user has to load a mesh file. To allow as many file types as possible, the model loading library Assimp [Ass06] is used. Once the mesh is imported, the user is presented with the screen shown in Figure 1 that displays the wireframe of the mesh that is color-coded by its normals. We use the approach by Gateau [Gat07] for rendering the wireframe. Since meshes are not always centered depending on the previous processing, they are moved to their geometric center point. Since the scale of the geometry is usually not saved in the mesh data file, the user has to select the scale of the current surface geometry, such that the proportions are correct for the subsequent simulation. In the side panel, additional information about the mesh is presented, such as triangle area, edge length, volume, and the bounding box. The colored circles below the surface geometry show the current progress of the pre-processing, with green indicating a finished step, yellow meaning that a step is ready to be performed, and red meaning it is not yet ready. This UI element enables the user to go back to any previous step without having to exit the program.

Inlet/Outlet Detection. In the next step, the user can either select the inlets and outlets manually or use the automatic detection. We provide a manual option to offer more control to the user, and in case the automatic fails to detect some inlets. Manual selection is realized through a texture look-up at the cursor position in an offscreen texture where ids of the currently visible surface vertices are stored. Then, the one-ring-neighborhood of the selected vertex is generated, though only vertices in the neighborhood whose normal is similar to the normal of the selected vertex are added. Therefore,



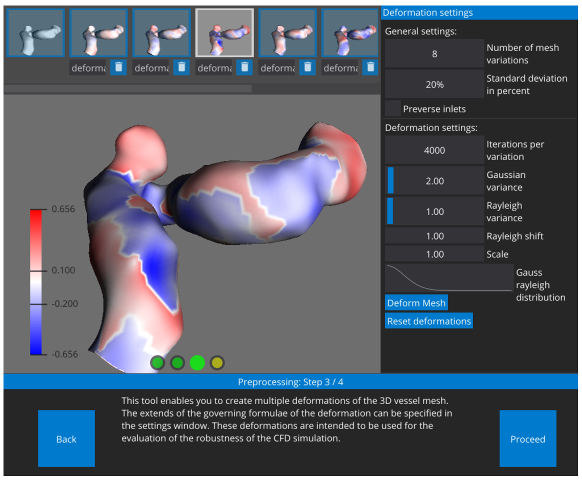
To select an inlet / outlet please press X on any vertex on the inlet / outlet.

Figure 3: *Selection process for the inlet, where the currently selected region is highlighted in red.*



Inlet / outlet count: 2

Figure 4: *Inlet/outlet viewer is displayed after selection.*



General settings:

8	Number of mesh variations
20%	Standard deviation in percent
	Preverse inlets

Deformation settings:

4000	Iterations per variation
2.00	Gaussian variance
1.00	Rayleigh variance
1.00	Rayleigh shift
1.00	Scale
	Gauss rayleigh distribution

This tool enables you to create multiple deformations of the 3D vessel mesh. The extends of the governing formulae of the deformation can be specified in the settings window. These deformations are intended to be used for the evaluation of the robustness of the CFD simulation.

Figure 5: *Mesh deformation. Above we show a list of the deformation results. After selecting an entry, it is displayed in the main window. The deformation options can be set in the right panel.*

we construct a half-edge data structure [BSBK02] of the surface mesh, as it allows easy access to the one-ring-neighborhood. This process is repeated for each vertex added to the neighborhood, until no more vertices with a similar normal can be reached this way. Figure 3 shows this mode, in which selected portions of the mesh are highlighted with a red wireframe color.

For the automatic detection of inlets and outlets, we implement a k-medoids clustering [KR87] on points that lie on a hard edge, *i.e.* whose adjacent triangles have normal vectors with an angle of at least 80 degrees. Additionally, the improved initial cluster seeding from k-means++ [AV07] is used. Then, we fit a plane to each cluster by applying the singular value decomposition [Van97], which enables us to find all points within the inlet/outlet. Once the inlets and outlets are found, we allow the user to reevaluate the detected vessel openings through an interactive list of all mesh parts, see Figure 4). Each inlet/outlet gets a specific color. Additionally, falsely detected vessel openings can be removed. By clicking on an inlet in the list, the camera is rotated to align to the surface normal of this inlet. Thereby, the user has an additional option to further differentiate between the individual inlets. The different parts of the mesh, such as all inlets or the vessel wall, can be hidden.

If the mesh is not capped at the inlets/outlets, these are automatically detected during mesh import. Here, the half-edge data structure is used, as it allows for easy detection of boundary loops by following the half-edges with no opposite edge until the starting edge is reached again.

Mesh Deformation. In the optional deformation step, a variable amount of mesh deformation can be generated. Figure 5 shows the available deformation settings as well as the result screen after the deformation is performed. In every iteration of the deformation process, a random location near or inside the mesh bounding box is selected, and the surface mesh points in its vicinity are moved according to the distribution. It is a combination of a Gaussian and a Rayleigh distribution with the formula:

$$f(x,\sigma_g^2,\sigma_r,s) = \frac{\exp^{-\frac{x^2}{2\sigma_g^2}}}{\sqrt{2\pi\sigma_g^2}} \cdot \frac{x+s}{\sigma_r} \exp^{-\frac{(x+s)^2}{2\sigma_r^2}}, \tag{1}$$

where σ_g^2 is the variance for the Gauss and σ_r^2 the variance for the Rayleigh distribution. s is a positive value that shifts the peak of the distribution, and x is the distance between the surface point and the position of the current deformation source. The combination of these two functions is used to allow for more customization options and to reduce the coverage of smoother Gaussian distributions, *i.e.* to flatten out the curve for higher x values. A flatter curve results in more global changes in the mesh, where larger regions are transformed (see case D in fig. 13), while a curve with a peak around $x = 0$ causes bigger changes in small regions which results in a noisy mesh surface (see case F and H in fig. 13). The user has direct control on the parameters of the distribution, namely the shift s, the variance for both the Gauss and the Rayleigh functions as well as a scaling factor that controls the strength of the deformation. The influence of these parameters on the curve is directly shown in the settings panel. The geometry of the inlets is vital for simulation stability. Therefore, we provide an option to prevent the inlets from getting deformed. If a standard deviation higher than 0% is set, the configuration values for each deformation are distributed within this standard deviation to include additional randomness. The results can be explored after the procedure is finished, see Figure 5. Here, the vertex-based distance to the original surface is mapped to a blue-to-red color map, with negative values denoting movement in the negative surface normal direction. In case a specific deformation is not in line with the user's requirements, it can be removed. Additionally, subsequent executions of the deformation procedure use the current deformation results as input. This way, the meshes can be deformed in a controlled iterative process.

Generate OpenFOAM Cases. At the end of the pre-processing, an OpenFOAM case is generated for each deformation as well as the original mesh. While every case is exported with its individual surface mesh, the general settings stay the same. Blood properties such as density and viscosity have to be specified, as they might differ depending on patient-specific information. The vessel inlet has to be defined, combined with the inflow velocity. However, the tool does not yet allow the use of velocity profiles, and therefore, transient simulations are not fully supported. To perform steady-state simulations, the `simpleFoam` utility is used with as many iterations as the user specifies. Additional settings that control the simulation, such as relaxation factors and tolerance, can be set, whereby the default settings are usually sufficient. For numerical schemes, we provide three preset options that should cover most applications. More complex scheme options are too specific and should, therefore, be edited in the exported `fvSchemes` dictionary. Parallel execution can also be enabled by specifying the count of subdomains in each axis direction that are solved in parallel.

The basic mesh generation settings include the surface mesh scale and the initial cell resolution of the domain. Since the simulation is calculated on grids, an initial subdivision of the domain has to be defined using the `blockMesh` utility, which needs the bounding box and a cell count per axis as input. The initial subdivision allows the mesh generation algorithm `snappyHexMesh` to perform further subdivision, following the configuration. This configuration is meant for more experienced users and contains most of the settings for this utility. The default settings are in line with suggestions found in the OpenFOAM documentation.

Some scalar values such as Lambda2 and WSS are vital in the assessment of rupture risk. Thus, these OpenFOAM post-processing options can be enabled here. The solver settings are useful for experienced users, as the default values are usually sufficient.

When exporting, files for each case are written to a separate directory within the user selected main directory. These files include the initial case conditions, all necessary OpenFOAM settings, and the individual surface meshes. Additionally, four bash scripts are generated that allow the user to run the simulations on a server cluster. The first bash script performs mesh generation, the second and third run the simulations, one sequentially, the other in parallel. The fourth script executes the post-processing utilities. All these scripts can be run from inside the tool.

4.2. Simulation

The simulation of blood flow in the vessel geometry using OpenFOAM consists of three steps, namely meshing, the simulation itself, and post-processing. In this section, we give an overview of the utilities used as well as our model assumptions for the experiments shown in Section 5.

Meshing. To generate an OpenFOAM volume mesh, several steps have to be performed, see Figure 6. First, an initial background grid of uniform hexahedral cells is created inside the bounding box of the blood vessel using `blockMesh`. Before the generated mesh is refined, the surface features are extracted from the input geometry using the `surfaceFeatureExtract` utility. Cells can then be split at feature edges, thereby preserving them in the computational domain to improve accuracy. The last meshing step is performed through the `snappyHexMesh` utility. First, a so-called castellated mesh is generated, where the cells of the background grid are split at the surface, and the feature edges created in the previous step. Then, cells that lie outside of the mesh are removed. Lastly, the castellated mesh is snapped to the surface, *i.e.*the grid points are moved onto to the surface in an iterative process.

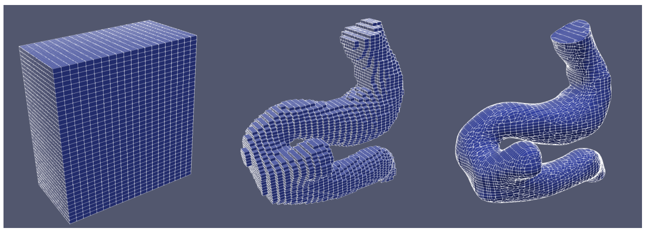


Figure 6: *OpenFOAM volume meshing: blockMesh (left), snappyHexMesh: castellated mesh step (middle), snappyHexMesh: snapping step (right).*

Mathematical Model Assumptions. We make some mathematical assumptions, which not have a considerable influence on the robustness of CFD overall but significantly improve simulation run-time. Since patient-specific velocities are unknown, we assume a fixed velocity magnitude of $0.5\,\mathrm{m/s}$ similar to velocity values specified in [JBS*15]. Thus, only steady state simulations are performed. In a study by Mantha *et al.* [MBHM09] flow patterns were found to persist throughout the cardiac cycle, which suggests that steady state simulations are a suitable alternative to unsteady simulations. Furthermore, we assume laminar flow and rigid walls. The flow is assumed to be Newtonian, which is usually a good approximation in regions with medium to high shear [JJCK04] and although the shear and the WSS could be overestimated in regions with slow flow, this is nevertheless useful in robustness evaluation.

CFD Solver. OpenFOAM provides several solver utilities that can be categorized by their algorithm. Since we employ steady-state simulations, we use the Semi Implicit Method for Pressure Linked Equations (SIMPLE) [VM07]. The *SIMPLE* algorithm is an iterative procedure that tries to find solutions for the pressure and the velocity field that both satisfy the momentum and the continuity equation, *i.e.* the Navier-Stokes equations. To run this in parallel, we use the specific OpenFOAM utilities. Parallel execution greatly reduces computation time even though the values of the unknown variables have to be synchronized between regions. Nonetheless, the simulations we performed took around one hour on average.

Post-Processing. Several variables are not calculated directly by the solver. We use the OpenFOAM post-processing utilities to calculate the WSS and two variables used for vortex detection, namely λ_2 and the Q-criterion.

4.3. Visualization of the Simulation Results

Once the simulation is finished, the results are imported into the visualization step of our tool. It is specifically designed for the use

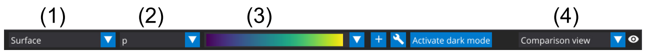


Figure 7: *Toolbar to select visualization* (1)*, pick calculated field* (2)*, select and add color maps* (3)*, and change current view* (4)*.*

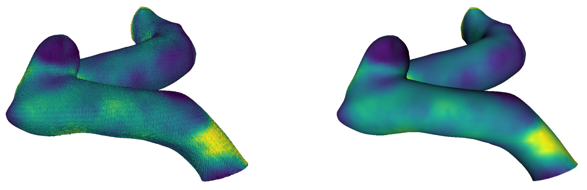
with OpenFOAM, facilitates a fast setup, and supports a comparative analysis, see Figure 1. Here, the user has to select two cases from a drop-down menu, which are then displayed side-by-side. The toolbar at the top provides global settings, see Figure 7. Various visualization techniques are available via a drop-down menu (1). The computed field that should be displayed can be chosen in the menu (2). The scalar values are color-coded using the selected color map, which can be chosen in the drop-down menu (3). The minimum value of the scalar field is mapped to the color on the far left and the maximum value to the color on the far right. However, these values can be changed by clicking the wrench icon, as some values may be too high or too low, thereby reducing distinguishability. The rightmost item in the bar (4) allows the user to select between two different views: a side-by-side comparison and a single display of each case.

The middle part of the layout, as shown in Figure 1, is used to display the blood vessel. To improve the ease of use of the split view, the camera between both sides is synchronized. Additionally, the layout consists of a play bar, an options menu, and a legend for the color-coded values. The play bar at the bottom allows the user to select specific time steps or iterations via the timeline or the step function. Furthermore, simulation results can be played with a small pause between the steps, such that each step is still recognizable. The gear icon in the lower left corner can be clicked to hide the options panel. Case-related options include cropping plane settings, Phong [Pho75] lighting model customization, time step export, and visualization-specific settings.

We integrated various visualizations that enable the display of both scalar and vector fields. As some fields are exclusively defined on the vessel surface or inside the vessel, visualizations for boundary and internal data as well as visualizations for a combination of both are provided. In Figure 8, the two implemented boundary visualizations are shown. The center points of boundary faces are rendered directly as sphere impostors, see Figure 8a. The sphere radius is specified as the length of the bounding box of the corresponding boundary face. In the case of impostors, an image of complex geometry is rendered instead of the geometry itself [BCF*05]. This visualization is also applicable for internal data where cell center points are used instead of boundary face center points.

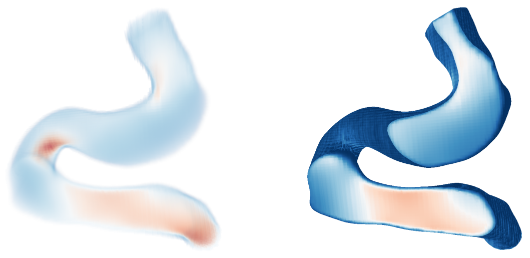
For the surface visualization shown in Figure 8b, the data has to be interpolated from the data points to the surface mesh vertices. To achieve reasonable performance when interpolating the data, references to data points, and interpolation weights from these points to each vertex are calculated and stored during case import. The data from these points is then interpolated onto the mesh vertices using a modified version of Shepard's interpolation method [She68]:

$$s(\mathbf{x}) = \begin{cases} \left(\Sigma_{i=0}^{N-1} w_i(\mathbf{x}) s_i\right) \div \left(\Sigma_{i=0}^{N-1} w_i(\mathbf{x})\right) & \text{if } d(\mathbf{x}, \mathbf{x_i}) \neq 0, \\ s_i & \text{otherwise} \end{cases} \qquad (2)$$



(a) *Boundary points* **(b)** *Surface*

Figure 8: *Boundary data visualizations comprising sphere imposters (left) and surface depiction (right).*



(a) *Clipped volume ray casting.* **(b)** *Volume to surface rendering with solid-clipping.*

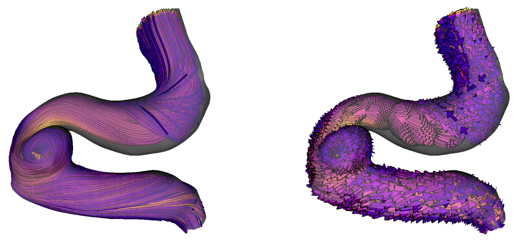
Figure 9: *Volume data visualizations.*

where $s(\mathbf{x})$ is the result of the interpolation for vertex $\mathbf{x}$, N is the number of data points, s_i is the scalar value at data point $\mathbf{x_i}$, and $d(\mathbf{x}, \mathbf{x_i})$ is the Euclidean distance function. The weight function $w_i(\mathbf{x})$ is taken from Franke and Nielson [FN80]:

$$w_i(\mathbf{x}) = \left(\frac{\max(0, R - d(\mathbf{x}, \mathbf{x_i}))}{R d(\mathbf{x}, \mathbf{x_i})} \right)^2 \qquad (3)$$

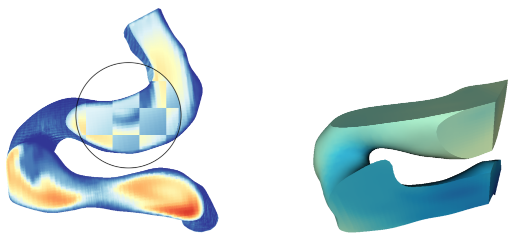
with R as the radius around the point $\mathbf{x}$ where data points are located that should have an influence on the interpolation.

To display volumetric data efficiently, we employ an OpenGL 3D volume texture. Scalar or vector values given at the data points are rendered into slices of the texture. The radius of the cell's bounding box determines the influence radius of the data point. We orientate the camera along the z-axis as the layers of 3D textures in OpenGL are oriented this way. Then, we can execute an OpenGL shader program on the point cloud. In the vertex shader, each point is associated with its scalar or vector value and its cell radius. The data can be interpolated by multiplying the radius with a factor greater than 1.0 and thereby increasing the influence radius of each data point to the data in the volume texture. As the extents of individual OpenFOAM cells might span multiple layers in the texture, a geometry shader is run with various invocations. Each invocation generates a slice in the shape of a circle such that the combination of the slices results in the sphere that represents the cell. The circle is then used as input to the fragment shader, which outputs the scalar or vector value of the cell. Additive blending is used to accumulate the values that are written into the texture. To determine the average, the values in the generated 3D texture are divided by the alpha value in a separate compute shader.



(a) *Streamlines.* **(b)** *Arrow glyphs.*

Figure 10: *Visualizations of vector fields.*



(a) *Case overlay with checkerboard pattern.* **(b)** *Clipped volume to surface visualization.*

Figure 11: *Additional visualization options.*

Figure 9 shows the two integrated volume visualization techniques for scalar data. We use volume ray casting to render the volume texture directly, see Figure 9a. Here, the bounding box of the surface mesh is used as a proxy geometry. The intersections between each view ray and the bounding box are calculated, and the volume is sampled along each ray starting at the front intersection until the back intersection is reached. To enable clipping in this visualization, the rays themselves are clipped. As a transfer function, we map the scalar values to alpha $= 0$ at minimum value and to alpha $= 1$ at maximum value with the selected color map. The method shown in Figure 9b utilizes the surface mesh to render the volume. At each vertex, a ray is generated in the negative direction of the surface normal. When stepping along this ray, the scalar value at the current position is read from the volume texture until a valid value is found or a fixed number of steps is reached. The main advantage of this visualization is that clipped surfaces are displayed as solid caps, which allows efficient volume slicing. These caps are rendered as planes using a method by Rossignac *et al.* [RMS92] and colored using the scalar data in the volume texture.

In addition to scalar field visualizations, flow patterns can be visualized, which might be related to aneurysm rupture. We provide both streamline (see Figure 10a) and glyph (see Figure 10b) visualizations to display the flow of the current velocity field. Each line is generated on the GPU starting from points on a seeding sphere computed using the Fibonacci lattice [Gon10] method. To improve usability, we display a sphere at the seeding sphere's position while seeding is inactive. For streamline generation, a compute shader is executed with these seed points and a volume texture with the current velocity field as input. For integration, the 4th-order Runge-Kutta scheme is used. If the velocity read from the texture is invalid, the streamline generation for this particular seed point is terminated. The tool provides a few settings to control the seeding as well as specific attributes such as line thickness and length or glyph size. Lines are rendered continuously, where a view-aligned quad represents each line segment. These line segments are also used to render arrow glyphs where the length of the glyphs is configurable. To improve the visual perception of lines, halos are rendered and shading, according to Zöckler *et al.* [ZSH96] is performed. Moreover, a path line visualization is provided to depict the time-dependent flow field. The positions of these particles, starting on a seeding sphere, have to be precomputed. For this purpose, the user can select a specific time step to start the path line integration.

Aside from the provided visualizations, there are a few other options to enable a more fine-grained comparison between cases as well as to improve visibility of data, see Figure 11. While a side-by-side comparison might be feasible in most cases, simultaneous exploration on the same view allows the user to more easily detect differences in scalar values and flow patterns. Therefore, a circular overlay shown in Figure 11a is integrated that provides a look at the current state of the other case. Inspired by Meuschke *et al.* [MGB*19], we provide this option to enable a screen-space checkerboard pattern inside the circle, which can be modified through user settings. Furthermore, up to 8 clipping planes can be configured according to the user. Thus, data inside the vessel that was previously occluded becomes visible, as shown in Figure 11b. Clipping planes can be added or removed, and both their position and their normal direction can be changed.

Since the files created by OpenFOAM during calculation are not in a universal format that can be imported by typical data analysis tools, an export functionality is provided in our tool. It allows the data from each field to be written to CSV files at the current time, enabling further data analysis outside our tool.

5. Experiments

To evaluate the robustness of CFD simulations, we applied our framework to four aneurysm cases, see Figure 12. For each case, we created eight variations of the original geometry. The volume differences between the original meshes and their deformations are presented in Table 1. In our analysis, we focus on the difference in WSS distribution and values for case 1 and 2, as well as differences in flow patterns for case 3 and 4.

5.1. Wall Shear Stress Analysis

First, we focus on WSS changes due to mesh deformation. As WSS is an essential factor for evaluating rupture risk, its robustness is essential in guaranteeing a correct assessment. Table 2 shows the average WSS values for each deformation of case 1 and 2.

Case 1. The aneurysm of case 1 is berry shaped with one inlet and one outlet. The generated OpenFOAM geometry has around 1700000 cells. When comparing the WSS values shown in Figure 13 with the volume changes, it is noticeable that higher volume coincides with lower WSS and vice versa (see geometry B, D, H).

	A	B	C	D	E	F	G	H
Case 1	2.90%	−0.65%	0.33%	6.13%	−0.45%	0.05%	4.50%	−3.97%
Case 2	1.96%	−5.02%	−1.71%	−2.61%	0.26%	−0.39%	−0.29%	0.67%
Case 3	8.12%	1.14%	−3.16%	−3.33%	3.54%	−4.82%	−0.21%	−0.34%
Case 4	−10.71%	−2.83%	5.60%	0.03%	−0.34%	0.14%	14.69%	3.19%

Table 1: *Volume difference between deformed and original geometries.*

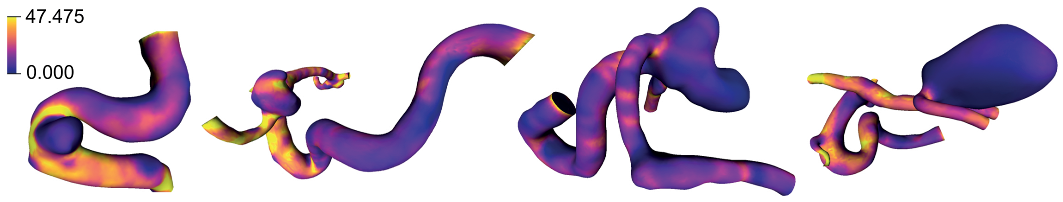


Figure 12: *Original meshes for all four cases color-coded according to WSS magnitude.*

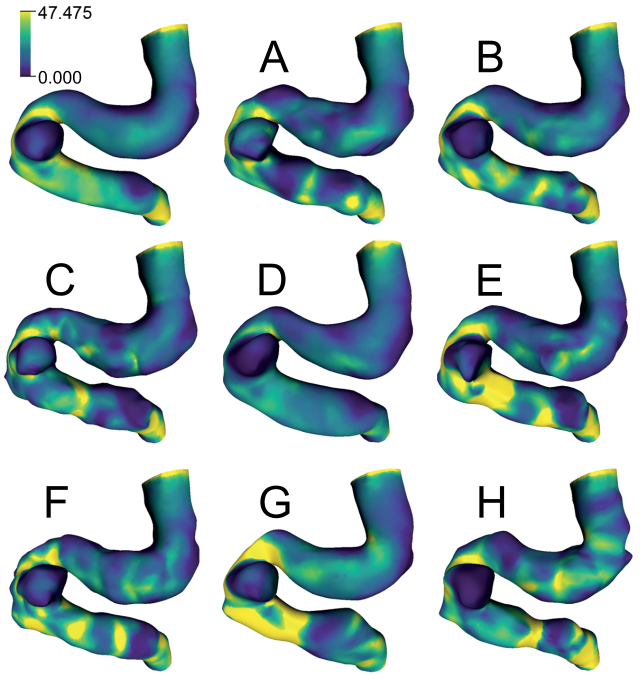


Figure 13: *Different deformations of the original mesh of case 1, which is shown in the top-left. The surface meshes show the WSS distribution with a scale from* 0 *to* 47.475*Pa.*

However, *e.g.* geometry A and G exhibit different behavior, possibly due to a higher amount of narrow regions in the mesh. Interesting are also the changes of WSS on the aneurysm for geometry A and C, whereas the overall WSS average is very similar compared to the original geometry. Due to the shape changes, there is a different area with increased WSS than in the original mesh, *e.g.* for A in the upper part of the aneurysm.

Case 2. Figure 14 shows the simulation results for case 2. The locations of elevated WSS are almost identical between deformations. However, for some deformations, *e.g.* E and G, there are small differences before and after artificial stenosis of the vessel. No direct correlation between volume differences and WSS average can be found in this case. The influence of stenosis might be higher than the impact of overall volume difference.

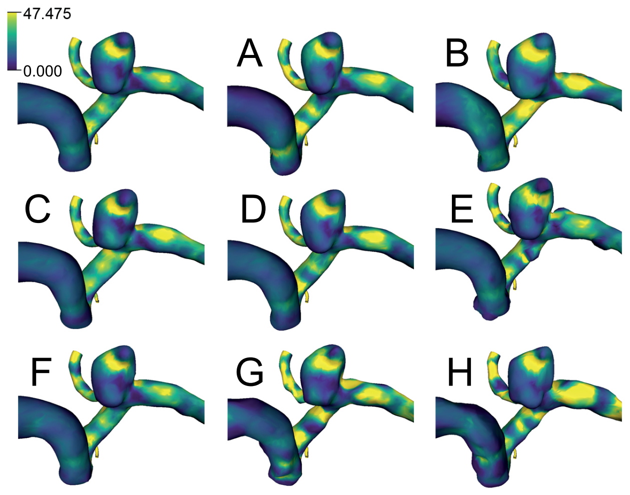


Figure 14: *Different deformations of the original mesh of case 2, which is shown in the top-left. The surface meshes show the WSS distribution with a scale from* 0 *to* 47.475*Pa.*

5.2. Flow Pattern Analysis

Intra-aneurysmal flow patterns might play a significant role in aneurysm rupture. Therefore, we evaluated the robustness of these patterns.

Case 3. Figure 15 shows a clipped view on the flow fields color-coded by the flow speed. The red circles indicate vortices. While the flow pattern in this cross-section shows only one vortex for the original geometry, for the geometries B, C, D, F, G and, H two or

	Original geometry	A	B	C	D	E	F	G	H
Case 1	19.76	20.27	20.59	19.40	14.73	21.10	19.73	26.86	23.85
Case 2	19.87	21.01	22.92	20.46	20.66	19.63	19.97	21.27	21.24

Table 2: *Average WSS values in Pa for cases* 1 *and* 2.

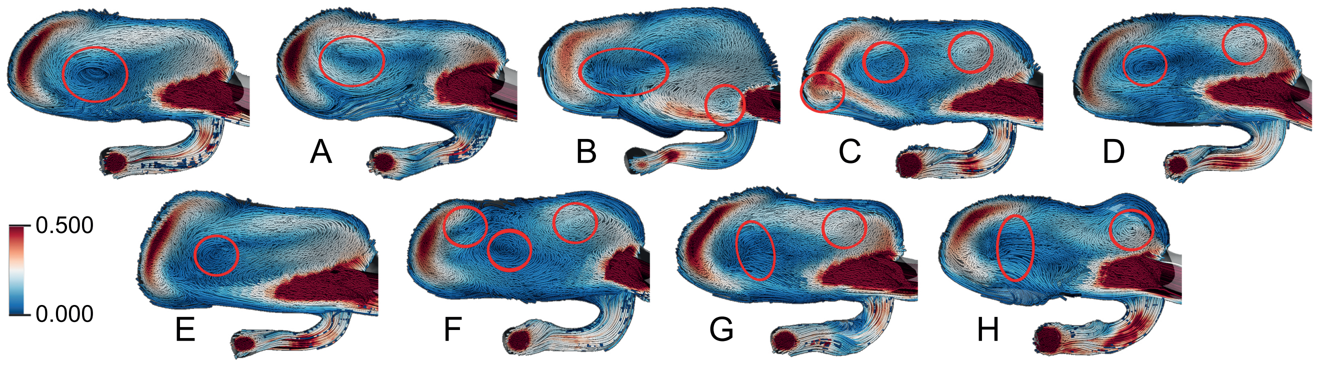


Figure 15: *Flow patterns inside the aneurysm different deformations of case 3. The streamlines show the velocity magnitude with a scale from* 0 *to* 0.5*m/s. Red circles are meant to emphasize regions of vortical flow.*

more vortices are visible. The volume difference does not seem to be an indicator of changing flow patterns, as geometry G and H have almost identical volume and geometry, while A has significantly higher volume compared to the original. Geometry A has, however, a similar aneurysm shape to the original, which might be a reason for similar flow patterns. Geometries G and H for example, both exhibit similar shape and flow pattern.

Case 4. The intra-aneurysmal flow patterns for case 4 are shown in Figure 16. The generated mesh for the original and each deformation consist of around 800000 cells. While there are some cases that have some similarities with the original, such as B and D, most geometries show different flow patterns. This is particularly remarkable for E and F, where the volume difference is small, and the shape is similar. Thus, there might be a different factor that influences the flow pattern. Mesh G is interesting, as well. Here, the overall increase in mesh volume results in lower flow rates, which in turn might cause less complex intra-aneurysmal flow.

6. Discussion

Our tool enables fast OpenFOAM case generation, including inlet detection and mesh deformation by providing a GUI on top of OpenFOAM. Additionally, we allow direct visualization of the simulation results, including efficient volume and streamline visualizations. Therefore, the overhead of the common OpenFOAM pipeline usage is greatly reduced.

While the deformation process is fast, it can only be applied to the complete mesh. The experiments show that even small geometry variations already result in significant changes in WSS magnitude as well as flow patterns. Some correlation was found between vessel volume and WSS magnitude, where lower volume results in higher WSS and vice versa. These findings coincide with findings from Berg *et al.* [BSV*18] who found differences in WSS values depending on the used reconstruction kernel. However, the mesh volume does not seem to be the only indicator. It is also important whether or not there are narrow vessel regions. These stenosis result in higher stress regardless of overall volume. In contrast, a more homogeneous vessel diameter seems to be an indication of both lower and more homogeneous WSS. The locations of elevated WSS, however, are mostly similar between geometric variations.

Moreover, the flow pattern analysis shows that even volume changes of less than 1% already result in different flow structure. For flow patterns, the driving factor is mostly the aneurysm shape rather than the volume difference between the meshes. As long as the aneurysm shape is similar between deformed cases, the flow pattern is mostly similar, while small shape differences might already result in more vortices and different patterns.

However, some cases contradict these findings. A possible reason might be that the flow patterns are also influenced by the shape of the ostium. Additionally, higher resolution volume meshes should be generated as some lower resolution cases present small instabilities. As no direct factor was found that indicated specific changes, it might be necessary to evaluate each case individually.

7. Conclusion and Future Work

We present an analysis framework to evaluate the robustness of CFD simulations w.r.t. variations in input geometry of cerebral aneurysms. Our tool covers the entire process consisting of data pre-processing, flow simulation, and visualization of the simulation results. For this purpose, several deformed variations of the input geometry are generated during pre-processing, imitating possible segmentation variations caused by manual or automatic segmentation procedures. Then, an OpenFOAM case is exported for each generated mesh as well as the initial mesh. After pre-processing, the OpenFOAM simulations are run, and the simulation results are

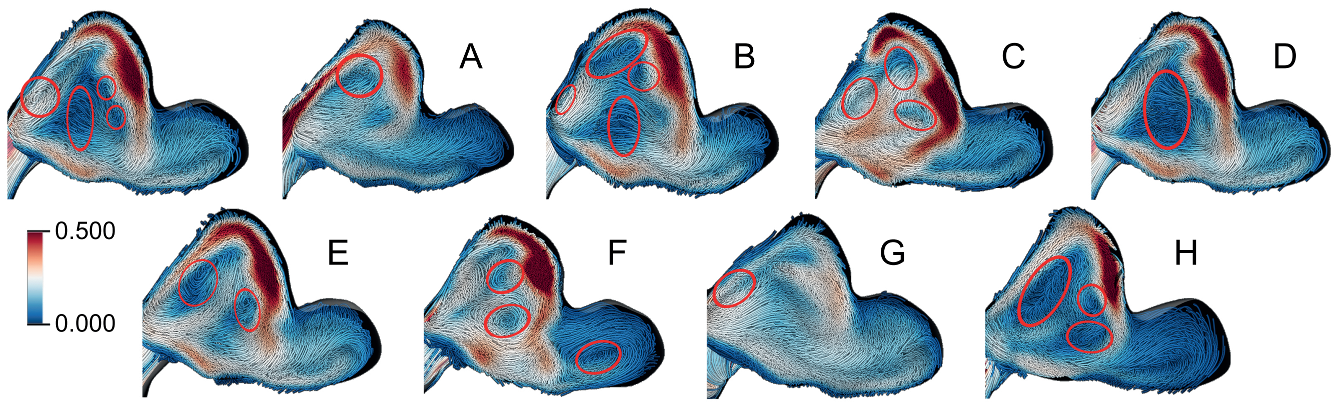


Figure 16: *Flow patterns inside the aneurysm different deformations of case 4. The streamlines show the velocity magnitude with a scale from* 0 *to* 0.5*m/s. Red circles are meant to emphasize regions of vortical flow.*

visualized using several techniques. The tool integrates neatly with OpenFOAM cases, and data import is seamless and fast.

Our results underline the need for accurate segmentation methods, as the robustness of CFD simulations to variations in mesh geometry appears to be reduced. Small shape differences can lead to significant differences in quantitative and qualitative flow properties. Unfortunately, no specific factor for the results could be isolated. The identification of these factors could provide valuable information on the robustness of CFD simulations in blood vessels and should, therefore, be further investigated in the future. For this purpose, the selection of the ostium should be added in the pre-processing phase. This would allow the calculation of aneurysmal volume change as well as further shape analysis. Additionally, more sophisticated mesh deformation options should be integrated to allow the selection of a specific deformation domain and to enable more complex deformations, *e.g.* generation of blebs on the aneurysm surface. It might also be beneficial to provide surface-aware manual deformation options to the user, such that they can directly control the deformation. To further improve the visual exploration of data, clustering methods could be added to reduce visual clutter, especially in the streamline view. A clustering approach could also enable the use of a difference measure on the streamlines between the flows in the original and the deformed mesh. The seeding process could also be improved by isolating the ostium and seeding the streamlines there. Furthermore, the tool would benefit from more sophisticated visualization techniques that show correlations between the deformation and the simulation results. This could be achieved for scalar data through *e.g.*the map-based approach proposed by Meuschke *et al.* [MVB*16]. Utilizing these improvements to the framework, a more in-depth analysis could be conducted that might isolate a direct factor.

References

[Ass06] ASSIMP TEAM. *The Open-Asset-Importer-Lib*. Accessed: 2019-06-10. 2006. URL: `http://www.assimp.org` 3.

[AV07] ARTHUR, DAVID and VASSILVITSKII, SERGEI. "k-means++: The advantages of careful seeding". *Proc. of the ACM-SIAM symposium on discrete algorithms*. J Soc Ind Appl Math. 2007, 1027–35 4.

[BCF*05] BEHRENDT, STEPHAN, COLDITZ, CARSTEN, FRANZKE, OLIVER, et al. "Realistic real-time rendering of landscapes using billboard clouds". *Comput Graph Forum*. Vol. 24. 2005, 507–16 6.

[BJT12] BERG, PHILIPP, JANIGA, GÁBOR, and THÉVENIN, DOMINIQUE. "Detailed comparison of numerical flow predictions in cerebral aneurysms using different CFD software". *Conference on Modelling Fluid Flow* (2012), 128–35 2.

[BSBK02] BOTSCH, MARIO, STEINBERG, STEPHAN, BISCHOFF, STEPHAN, and KOBBELT, LEIF. "OpenMesh–a generic and efficient polygon mesh data structure". *OpenSG Symposium*. 2002 4.

[BSN06] BRISMAN, JONATHAN L, SONG, JOON K, and NEWELL, DAVID W. "Cerebral aneurysms". *N Engl J Med* 355.9 (2006), 928–39 1.

[BSV*18] BERG, P, SAALFELD, S, VOSS, S, et al. "Does the DSA reconstruction kernel affect hemodynamic predictions in intracranial aneurysms? An analysis of geometry and blood flow variations". *J Neurointerv Surg* 10.3 (2018), 290–6 2, 9.

[CC15] CHUNG, BONGJAE and CEBRAL, JUAN RAUL. "CFD for Evaluation and Treatment Planning of Aneurysms: Review of Proposed Clinical Uses and Their Challenges". *Ann Biomed Eng* 43.1 (2015), 122–38 2.

[CMG11] CURRIE, S., MANKAD, K., and GODDARD, A. "Endovascular treatment of intracranial aneurysms: review of current practice". *Postgrad Med J* 87.1023 (2011), 41–50 1.

[FN80] FRANKE, RICHARD and NIELSON, GREG. "Smooth interpolation of large sets of scattered data". *Int J Numer Meth Eng* 15.11 (1980), 1691–704 6.

[Gat07] GATEAU, SAMUEL. "Solid wireframe". *NVIDIA Whitepaper WP-03014-001 v01* (2007) 3.

[GNBP11] GASTEIGER, ROCCO, NEUGEBAUER, MATHIAS, BEUING, OLIVER, and PREIM, BERNHARD. "The FLOWLENS: A focus-and-context visualization approach for exploration of blood flow in cerebral aneurysms". *IEEE Trans Vis Comput Graph* 17.12 (2011), 2183–92 2.

[Gon10] GONZÁLEZ, ÁLVARO. "Measurement of areas on a sphere using Fibonacci and latitude–longitude lattices". *Math Geosci* 42.1 (2010), 49 7.

[GSK*11] GOUBERGRITS, LEONID, SCHALLER, JENS, KERTZSCHER, ULRICH, et al. "Statistical wall shear stress maps of ruptured and unruptured middle cerebral artery aneurysms". *J Royal Soc Interface* 9.69 (2011), 677–88 2.

[JBS*15] JANIGA, G., BERG, P., SUGIYAMA, S., et al. "The Computational Fluid Dynamics Rupture Challenge 2013—Phase I: Prediction of Rupture Status in Intracranial Aneurysms". *Am J Neuroradiol* 36.3 (2015), 530–6 5.

[JJCK04] JOHNSTON, BARBARA M., JOHNSTON, PETER R., CORNEY, STUART, and KILPATRICK, DAVID. "Non-Newtonian blood flow in human right coronary arteries: steady state simulations". *J Biomech* 37.5 (2004), 709–20 5.

[KLG*11] KUHN, ALEXANDER, LEHMANN, DIRK J, GASTEIGER, ROCCO, et al. "A Clustering-based Visualization Technique to Emphasize Meaningful Regions of Vector Fields." *VMV*. 2011, 191–198 2.

[KR87] KAUFMAN, LEONARD and ROUSSEEUW, PETER. *Clustering by means of medoids*. North-Holland, 1987 4.

[KW03] KRUGER, JENS and WESTERMANN, RÜDIGER. "Acceleration techniques for GPU-based volume rendering". *Proc. of the IEEE Visualization '03*. IEEE Computer Society. 2003, 38 2.

[LC87] LORENSEN, WILLIAM E. and CLINE, HARVEY E. "Marching Cubes: A High Resolution 3D Surface Construction Algorithm". *Proc. of Conference on Computer Graphics and Interactive Techniques*. ACM, 1987, 163–9 3.

[LGP14] LAWONN, KAI, GASTEIGER, ROCCO, and PREIM, BERNHARD. "Adaptive surface visualization of vessels with animated blood flow". *Comput Graph Forum*. Vol. 33. 2014, 16–27 2.

[MBHM09] MANTHA, AISHWARYA R, BENNDORF, GOETZ, HERNANDEZ, ANDRES, and METCALFE, RALPH W. "Stability of pulsatile blood flow at the ostium of cerebral aneurysms". *J Biomech* 42.8 (2009), 1081–7 5.

[MGB*19] MEUSCHKE, MONIQUE, GÜNTHER, TOBIAS, BERG, PHILIPP, et al. "Visual Analysis of Aneurysm Data using Statistical Graphics". *IEEE Trans Vis Comput Graph* 25.1 (2019), 997–1007 3, 7.

[MNP11] MOENCH, TOBIAS, NEUGEBAUER, MATHIAS, and PREIM, BERNHARD. "Optimization of vascular surface models for computational fluid dynamics and rapid prototyping". *Int. Workshop on Digital Engineering*. 2011, 16–23 3.

[MOB*18] MEUSCHKE, MONIQUE, OELTZE-JAFRA, STEFFEN, BEUING, OLIVER, et al. "Classification of Blood Flow Patterns in Cerebral Aneurysms". *IEEE Trans Vis Comput Graph* (2018) 2.

[MTHG03] MATTAUSCH, OLIVER, THEUSSL, THOMAS, HAUSER, HELWIG, and GRÖLLER, EDUARD. "Strategies for interactive exploration of 3D flow using evenly-spaced illuminated streamlines". *Proc. of the Spring Conference on Computer Graphics*. ACM Press, 2003, 213 2.

[MVB*16] MEUSCHKE, MONIQUE, VOSS, SAMUEL, BEUING, OLIVER, et al. "Combined visualization of vessel deformation and hemodynamics in cerebral aneurysms". *IEEE Trans Vis Comput Graph* 23.1 (2016), 761–70 10.

[NSA*09] NIEUWKAMP, DENNIS J, SETZ, LARISSA E, ALGRA, ALE, et al. "Changes in case fatality of aneurysmal subarachnoid haemorrhage over time, according to age, sex, and region: a meta-analysis". *Lancet Neurol* 8.7 (2009), 635–42 1.

[OLK*14] OELTZE, STEFFEN, LEHMANN, DIRK J, KUHN, ALEXANDER, et al. "Blood flow clustering and applications invirtual stenting of intracranial aneurysms". *IEEE Trans Vis Comput Graph* 20.5 (2014), 686–701 2.

[PB13] PREIM, BERNHARD and BOTHA, CHARL P. *Visual computing for medicine: theory, algorithms, and applications*. Newnes, 2013 2.

[Pho75] PHONG, BUI TUONG. "Illumination for computer generated pictures". *Commun ACM* 18.6 (1975), 311–7 6.

[RMS92] ROSSIGNAC, JAREK, MEGAHED, ABE, and SCHNEIDER, BENGT-OLAF. "Interactive inspection of solids: cross-sections and interferences". *ACM SIGGRAPH Computer Graphics*. Vol. 26. ACM. 1992, 353–60 7.

[Sch97] SCHÖBERL, JOACHIM. "NETGEN An advancing front 2D/3D-mesh generator based on abstract rules". *Comput Vis Sci* 1.1 (1997), 41–52 3.

[She68] SHEPARD, DONALD. "A Two-dimensional Interpolation Function for Irregularly-spaced Data". *Proc. of the ACM National Conference*. ACM, 1968, 517–24 6.

[SMA*13] SCHNEIDERS, JJ, MARQUERING, HA, ANTIGA, L, et al. "Intracranial aneurysm neck size overestimation with 3D rotational angiography: the impact on intra-aneurysmal hemodynamics simulated with computational fluid dynamics". *Am J Neuroradiol* 34.1 (2013), 121–8 2.

[SQAM14] SEN, YUKA, QIAN, YI, AVOLIO, ALBERTO, and MORGAN, MICHAEL. "Image segmentation methods for intracranial aneurysm haemodynamic research". *J Biomech* 47.5 (2014), 1014–9 2, 3.

[Van97] VAN HUFFEL, SABINE. *Recent advances in total least squares techniques and errors-in-variables modeling*. Vol. 93. SIAM, 1997 4.

[VBS*18] VALEN-SENDSTAD, KRISTIAN, BERGERSEN, ASLAK W., SHIMOGONYA, YUJI, et al. "Real-World Variability in the Prediction of Intracranial Aneurysm Wall Shear Stress: The 2015 International Aneurysm CFD Challenge". *Cardiovasc Eng Techn* 9.4 (2018), 544–64 2.

[VM07] VERSTEEG, HENK KAARLE and MALALASEKERA, WEERATUNGE. *An introduction to computational fluid dynamics: the finite volume method*. Pearson education, 2007 5.

[VS14] VALEN-SENDSTAD, KRISTIAN and STEINMAN, DAVID A. "Mind the gap: impact of computational fluid dynamics solution strategy on prediction of intracranial aneurysm hemodynamics and rupture status indicators". *Am J Neuroradiol* 35.3 (2014), 536–43 2.

[Wie03] WIEBERS, DAVID O. "Unruptured intracranial aneurysms: natural history, clinical outcome, and risks of surgical and endovascular treatment". *Lancet* 362.9378 (2003), 103–10 1.

[WP11] WONG, GEORGE KC and POON, WS. "Current status of computational fluid dynamics for cerebral aneurysms: the clinician's perspective". *J Clin Neurosci* 18.10 (2011), 1285–8 2.

[XTK*12] XIANG, JIANPING, TREMMEL, MARKUS, KOLEGA, JOHN, et al. "Newtonian viscosity model could overestimate wall shear stress in intracranial aneurysm domes and underestimate rupture risk". *J Neurointerv Surg* 4.5 (2012), 351–7 2.

[ZSH96] ZÖCKLER, MALTE, STALLING, DETLEV, and HEGE, H-C. "Interactive visualization of 3D-vector fields using illuminated stream lines". *Proc. of Visualization'96*. IEEE. 1996, 107–13 7.