Technical report

Nr.: FIN-06-2012

Evaluation of Streamline Clustering Techniques for Blood Flow Data

S. Oeltze, D.J. Lehmann, H. Theisel, B. Preim

Arbeitsgruppe Visualisierung

Fakultät für Informatik
Otto-von-Guericke-Universität Magdeburg

Nr.: FIN-06-2012

Evaluation of Streamline Clustering Techniques for Blood Flow Data

S. Oeltze, D.J. Lehmann, H. Theisel, B. Preim

Arbeitsgruppe Visualisierung

Technical report (Internet)
Elektronische Zeitschriftenreihe
der Fakultät für Informatik
der Otto-von-Guericke-Universität Magdeburg
ISSN 1869-5078

Fakultät für Informatik
Otto-von-Guericke-Universität Magdeburg

Impressum (§ 5 TMG)

Herausgeber:
Otto-von-Guericke-Universität Magdeburg
Fakultät für Informatik
Der Dekan

Verantwortlich für diese Ausgabe:
Otto-von-Guericke-Universität Magdeburg
Fakultät für Informatik
Steffen Oeltze
Postfach 4120
39016 Magdeburg
E-Mail: stoeltze@isg.cs.uni-magdeburg.de

http://www.cs.uni-magdeburg.de/Technical_reports.html

Technical report (Internet)
ISSN 1869-5078

Redaktionsschluss: 17.12.2012

Bezug: Otto-von-Guericke-Universität Magdeburg
Fakultät für Informatik
Dekanat

Evaluation of Streamline Clustering Techniques for Blood Flow Data

Steffen Oeltze, Dirk J. Lehmann, Holger Theisel, Bernhard Preim

Abstract—Understanding the hemodynamics of blood flow in vascular pathologies such as aneurysms is essential for both their diagnosis and treatment. Computational fluid dynamics (CFD) simulations of blood flow based on patient-individual data are performed to better understand aneurysm initiation and progression and for predicting treatment success. A CFD simulation results in a complex, multiparameter dataset comprising scalar as well as vectorial data attributes. For its comprehensive investigation, the contained flow information is often visualized by a highly dense and cluttered set of integral curves colored according to one of the attributes.
We aim at a fully automatic approach for reducing visual clutter and exposing characteristic flow structures by grouping similar curves and computing group representatives. In this work, we lay the foundations by evaluating different clustering techniques for grouping curves. We evaluate *Spectral Clustering* and four versions of *Agglomerative Hierarchical Clustering*. Both are particularly suited since they can be based on inter-curve distances rendering the construction of feature vectors unnecessary. Our work focuses on steady-state simulations of blood flow in intracranial aneurysms and the visualization by means of streamlines. Our results indicate that Spectral Clustering as well as Agglomerative Hierarchical Clustering with average link or Ward's method as proximity measure generate meaningful groups of similar streamlines.

1 Introduction

Intracranial aneurysms, also referred to as cerebral or brain aneurysms, represent a pathologic balloon like dilation of cerebral vasculature due to a weakness of the arterial wall. They occur with a prevalence of about 2% in Western Europe [1]. They bear a high risk for the patient, since their rupture is associated with a mortality rate of $\approx 50\%$. In recent research, CFD simulations, which generate patient-specific hemodynamic data, are employed in assessing the risk of aneurysm rupture [2], [3], [4] and in predicting the success of different treatment options, e.g., stenting [5], [6], [7]. The results of these simulations are highly complex, multiparameter datasets comprising several scalar as well as vectorial attributes. For their comprehensive investigation, the contained flow information is often visualized by a dense and cluttered set of integral curves colored according to one of the attributes, e.g., velocity magnitude or pressure. The underlying blood flow pattern is then investigated by CFD engineers via manually reducing visual clutter in a tedious iterative procedure of selectively hiding and showing curves.

We aim at a fully automatic approach for reducing visual clutter and exposing characteristic structures of the flow by grouping curves and computing group representatives. In this work, we lay the foundation by comparing different clustering techniques for grouping. We investigate *Spectral Clustering* and four versions of *Agglomerative Hierarchical Clustering*: single link, complete link, average link, and Ward's method. Spectral Clustering and Agglomerative Hierarchical Clustering are particularly suited in this context since they can be based on inter-curve distances rendering the construction of feature vectors unnecessary. They have been applied before to the clustering of streamlines, e.g., in engineering or climate data [8], [9], [10], and for the grouping of fibers tracts extracted from Diffusion Tensor Imaging data [11], [12]. In our work, we focus on steady-state simulations of blood flow and its visualization by means of streamlines. It has been shown that major aspects of aneurysmal hemodynamics may be inferred from steady flow, which is faster and easier to compute than pulsatile flow [13].

In the clustering of blood flow, the number of clusters in unknown. Hence, we extend each clustering technique by a state-of-the-art method, which automatically determines a reasonable number. We apply all techniques to ten datasets and systematically investigate the results. This investigation identifies one method for automatically determining a reasonable number of clusters k as particularly reliable. Hence, all techniques are reapplied but this time based on k. In each clustering result, a unique cluster label is assigned to each streamline. Since no ground truth exist in the form of external labels, e.g., created by an expert, we evaluate the results by means of selected internal cluster indices. The evaluation indicates that three of the tested techniques generate meaningful groups of streamlines.

2 Technical Background

This section familiarizes the reader with the hemodynamic data generation pipeline, starting with the acquisition of patient image data and ending with a CFD simulation of blood flow. The pipeline has been described in detail by Gasteiger et al. [14] and is briefly summarized in the following. In the first step, image data of the aneurysm morphology including the vasculature in the close surrounding is acquired, e.g., by 3D rotational angiography. Next, the aneurysm and the surrounding vasculature are segmented. Thresholding techniques are feasible due to the high vessel-to-tissue contrast. Afterwards, a surface mesh is reconstructed from the segmentation result by Marching Cubes. The resulting mesh is then optimized with respect to mesh quality [15]. Then, the *ostium* is extracted as

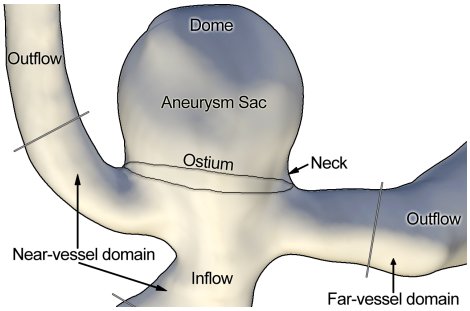


Fig. 1. Morphological features of a saccular cerebral aneurysm and subdivision of the surrounding vascular domain.

an anatomical landmark (Fig. 1) [16]. It is frequently used by domain experts to explore the flow into the aneurysm, e.g., by seeding streamlines there [17]. Finally, a volume mesh is required in order to perform the numerical simulation. All volume meshes have been generated based on the surface meshes using the commercial tool ANSYS IcemCFD (Ansys Inc., Canonsburg, PA, U.S.). The obtained grid quality has been carefully checked and maintained within the optimal range as needed for a successful computation. The generated body-fitting meshes involve up to 13,500,000 finite volume cells (tetrahedra).

Fluid flow simulations have been performed using the commercial software ANSYS Fluent 12 (Ansys Inc., Canonsburg, PA, U.S.). This code solves the governing equations of a fluid flow problem discretized using the finite volume method. The numerical simulation retains steady state computations using a Newtonian description with constant density and viscosity for the blood. The Reynolds numbers are in the laminar regime, proving that laminar solution is acceptable here. At the inlet of the computational domain a fully developed velocity profile is prescribed. The vessel walls are considered rigid during the computation. A standard, no-slip boundary condition is applied at all contact points with surfaces including the vessels. At all the outlets traction-free conditions have been employed. Computations are carried out in parallel using up to eight Linux computing cores (2.1 GHz AMD Opteron 64-bit dual-quad processors). For the finest mesh considered in this study, 19.5 GB of computer memory are needed for the accurate double-precision simulation.

3 Related Work

The huge variety of flow visualization techniques has been categorized by Post et al. [18] into direct, texture-based, geometric, and feature-based techniques. Salzbrunn et al. [19] added the class of partition-based flow visualization into which our work fits best. Partition-based techniques decompose a flow field based on vector values, integral curve properties or contained features. We briefly recapitulate approaches that are closely related to our work in the sense that they partition the flow based on integral curves. Some of the approaches use representatives to reduce visual clutter and highlight important flow structures. We classify the approaches into user-defined partitioning as well as automatic partitioning.

3.1 User-guided Integral Curve Clustering

The approaches in this class investigate the vector field by stream- or pathlines and decompose the set of lines according to a user-defined behavior. Salzbrunn and Scheuermann [20] proposed combined Boolean predicates based on predefined scalar quantities which determine for each streamline whether it has a desired property or not. Predicates on pathlines have recently been applied to the visual analysis of measured blood flow in aortic aneurysms [21]. A residence time predicate has been used as a tool for assessing the risk of blood clot development. Shi et al. [22] proposed a set of local and global attributes, e.g., curvature and Lyapunov exponent, which are computed for pathlines and the resulting scalar fields are inspected using a visual analytics approach. Pobitzer et al. [23] recently demonstrated for five different datasets the application of a statistics-based dimension reduction to the set of attributes in order to detect relevant, independent attributes. Two other approaches let the user specify interesting integral curves or curve parts in the observation space instead of the attribute space. In [24], a new concept is presented for virtually probing measured cardiovascular flow avoiding a tedious segmentation of the vessels. Streamlines and pathlines may be seeded from the probing geometry for inspecting the flow. Gasteiger et al. [14] employ a lens metaphor for generating focus-and-context visualizations. Streamline parts may be highlighted or attenuated inside the lens depending on the specific application task.

3.2 Automatic Integral Curve Clustering

Our work is strongly related to this class of approaches, which automatically cluster the set of integral curves into meaningful groups. The approaches differ by the algorithm that is applied for grouping and by the similarity measure steering which curves should be grouped. Bidmon et al. [25] propose a specialized algorithm for clustering the paths of solvent molecules in molecular dynamics simulations. The applied similarity measure considers dynamic attributes of the molecules as well as their pathways. An entropy-based streamline seeding strategy is followed by a two-step k-means clustering in [26]. The first step considers only the coordinates of the endpoints and the midpoint of each streamline to generate a coarse clustering. This is then refined by considering vector and shape properties in the second step. In both steps, Euclidean distance is applied as a similarity measure. Representatives for the clusters are computed by determining the streamlines that are closest to the centroids computed by k-means. They are visualized by streamtapes rendered in an illustrative fashion. In [8], Agglomerative Hierarchical Clustering (AHC) with average link has been used for grouping streamlines and pathlines. The authors put special emphasis on a new similarity measure that facilitates an interactive, cluster-based exploration of flow with streamline seeding rakes (standard point-wise Euclidean distance tests are computationally expensive and prevent such interactivity). The resulting streamlines may be pruned on a per-cluster basis by thresholding the new similarity measure. A saliency-guided streamline seeding is followed by AHC with single link in

[10]. Since single link is prone to the *chaining-effect* and outliers, the resulting clustering tree requires a subsequent top-down balancing. Instead of showing lines as representatives that are close to cluster centroids, the cluster boundaries are extracted and boundary streamlines are displayed. Rössl and Theisel [9] discuss the theory of a spectral embedding of streamlines. They demonstrate *Spectral Clustering* (SC) in the embedding space and compare various similarity measures in this process. The clustering result is visualized by coloring all streamlines according to their cluster ID. Very similar to the clustering of integral curves is the clustering of fiber tracts extracted from diffusion tensor MRI (DTI) data. Rössl and Theisel adopt their tested similarity measures from this domain. White matter fiber tracts are partitioned by means of specialized SC approaches in [12] and [27]. Three methods for determining cluster representatives are tested in [28]. Moberts et al. compare AHC with different links and different similarity measures for clustering fiber tracts [11]. A new similarity measure for fiber tracts in conjunction with AHC using single link is introduced in [29].

Our literature review identifies AHC and SC as the most widely used algorithms for automatically clustering integral curves and fiber tracts. Hence, we evaluate both techniques in the context of blood flow. In this process, we extend both techniques by a state-of-the-art approach for estimating a meaningful number of clusters.

4 Input for Streamline Clustering

In this section, we provide details on streamline generation, their properties and on the definition of streamline similarity.

4.1 Streamline Generation

The input for the streamline generation is the volume mesh from the simulation (Sec. 2). The mesh is represented as an unstructured grid composed of tetrahedral cells. A vector is stored at each cell point. Before the streamlines are generated, the mesh is manually cropped such that it contains only the aneurysm and the so-called *near-vessel domain* (Fig. 1). This is inspired by the work of Cebral et al. and Shojima et al. [17] and [30]. It enables us to focus the analysis on the aneurysm. The expressiveness of the clustering is strongly improved by this procedure. Assume that the similarity of streamlines is computed based on their geometry. It is very likely that streamlines follow a similar course in the inflow region and they may also follow a similar course in an outflow region. However, depending on where they enter the aneurysm, their course may strongly differ inside. If the *far-vessel domain* was now also considered in clustering, this effect would have less impact.

In order to assess the in- and outflow of the aneurysm, we seed streamlines at the ostium. The ostium is given as a triangle mesh. The mesh has been optimized for streamline seeding such that its vertices are homogeneously distributed [16]. Thus, the under- and overrepresentation of flow parts is avoided. The number of vertices is a parameter of the meshing algorithm. It is adjusted until the resulting surface well resembles the original vessel wall. Furthermore, it has been visually validated that the chosen number of vertices leads to streamlines that capture all relevant flow structures. If this was not the case, the meshing was repeated creating the doubled number of vertices. Since this was a subjective and time-consuming process, we rather used a higher number of vertices.

Streamlines have then been generated using the free software ParaView (Kitware, Clifton Park, NY, U.S.). A 5th order Runge-Kutta method has been employed with an integration step size that is automatically adjusted according to an estimated error. The integration has been carried out on the flow field in backward and forward direction. The resulting two streamlines have then been merged such that a linear traversal of the vertices from start to end is possible.

4.2 Streamline Properties and Similarity

The generated streamlines differ in their number of points as well as in their length. The number of points depends on the integration step size which is constantly updated by the employed algorithm. While it has an impact on the computational time of inter-streamline similarity calculation, it does not influence the similarity itself. In contrast, streamline length has a strong impact. Two streamlines may follow a very similar course for a long time but then, one of them is terminated. In most similarity measures, a much higher weight is assigned now to the difference in length then to the similarity over a long run. In all our datasets, a few lines follow a course very similar to a large set of neighboring lines but are considerably shorter. They occur close to the vessel wall due to early termination of the integration. We consider them as incomplete rather than incorrect data entities. Instead of separating them, the clustering should group them with the streamlines having a similar course.

The determination of streamline similarity is a prerequisite for streamline clustering. Similarity if often expressed by a distance measure. The choice of a measure depends on the application. General requirements are that it must be positive-definite and symmetric. A valid example in the context of streamlines is the Hausdorff distance which is based on streamline geometry. However, this distance is very sensitive to streamline length, since it outputs the maximum of point-wise distances [9]. A less sensitive measure referred to as *mean of closest point distances* (MCPD) is proposed in [31]:

$$d_M(s_i,s_j) = mean(d_m(s_i,s_j), d_m(s_j,s_i)) \qquad (1)$$
$$with\ d_m(s_i,s_j) = mean_{p_l \in s_i} \min_{p_k \in s_j} \|p_k - p_l\|$$

MCPD has been successfully employed for clustering fiber tracts and streamlines [11], [10], [9]. Hence, we adopted MCPD and applied it to blood flow clustering. Initial tests showed good results but also revealed that MCPD is still too sensitive to differences in streamline length, in particular when being used with versions of Agglomerative Hierarchical Clustering that are sensitive to outliers (Tab. 1). Very small-sized, outlier-corrupted clusters are generated whose representatives cause a distorted flow summary visualization. In order

to further reduce the sensitivity of MCPD to length, we replace the outer mean in Equation 1 by a minimum computation:

$$d_M(s_i,s_j) = min(d_m(s_i,s_j),d_m(s_j,s_i)) \quad (2)$$
$$with\ d_m(s_i,s_j) = mean_{p_l \in s_i} \min_{p_k \in s_j} \|p_k - p_l\|$$

If two lines follow a very similar course and then, one is terminated while the other one continuous, the distance from the shorter line to the longer one is now chosen as their inter-streamline distance. This results in a high similarity increasing the chance of being grouped together.

5 Streamline Clustering Techniques

Our literature review in Section 3 identified Agglomerative Hierarchical Clustering and Spectral Clustering as the most widely used algorithms for automatically clustering integral curves and fiber tracts. One reason is their ability to cluster data based on distances without the previous definition of feature vectors. Other popular algorithms, such as k-means, require the definition of a feature space. The space may be spanned by the line point coordinates which however, requires a resampling of the lines to a uniform number of points. An alternative way is to derive features, e.g., curvature or length, from the lines. Nevertheless, this requires an additional description of spatial similarity or dissimilarity for instance by combining features and coordinates. Each streamline would then be represented by a feature vector comprising coordinates and possibly derived features.

In the following, we describe *Agglomerative Hierarchical Clustering* and *Spectral Clustering*. Since the number of clusters is unknown in blood flow data, we combine each algorithm with a state-of-the-art technique for automatically computing a reasonable number of clusters. This computation helps us in providing a good initial visual summary of the flow, it aims at a more standardized comparison of the flow in a stented and an untreated aneurysm, and it supports a categorization of a large database of aneurysms that is planned for the future.

5.1 Agglomerative Hierarchical Clustering

The following explanation of Agglomerative Hierarchical Clustering (AHC) is based on [32]. The algorithm starts with each streamline as a cluster and then, repeatedly merges the two closest clusters until a single cluster remains. The merge step relies on a distance matrix $\mathbf{M}$ and a measure of cluster proximity. In our case, the squared, symmetric distance matrix contains all pair-wise inter-streamline distances computed according to Equation 2. Various cluster proximity measures have been published among which *single link*, *complete link*, *average link*, and *Ward's method* are the most popular ones. In single link, the proximity of two clusters is defined as the minimum distance between any two points in the different clusters. This approach can handle clusters of arbitrary shape, it tolerates considerable differences in cluster size but it is sensitive to outliers. Furthermore, it is infamous for the *chaining effect* leading to clusters containing very dissimilar elements which are connected by a chain of similar elements via some transitive relationship. In complete link, the proximity of two clusters is computed as the maximum distance between any two points in the different clusters. Complete link is less susceptible to outliers but tends to break large clusters and it favors globular cluster shapes. Average link is an intermediate approach between single and complete link. It also strives for globular compact clusters [33]. Here, the proximity of two clusters is defined as the average proximity between pairs of points in the different clusters. Ward's method aims at minimizing the total within-cluster variance. It defines the proximity of two clusters as the sum of squared distances between any two points in the different clusters (SSE: sum of squared errors). Due to the SSE-based proximity, Ward's methods favors globular cluster shapes. It was shown to prefer clusters with similar size and to be robust against outliers in the context of 2D curves [34].

In summary, all versions of AHC lack a global objective function that shall be optimized. Instead, they decide locally which clusters should be merged. The merging decisions cannot be undone such that bad decisions, i.e., involving outliers, are propagated throughout the entire clustering process. A strength of AHC is the ability to rapidly generate results based on different numbers of clusters by simply cutting the cluster tree at different levels. Furthermore, it is non-parametric except for the choice of a proximity measure. Both strengths explain its frequent use in applications where the "correct" number of clusters is unknown. In such a case, the user often sequentially browses through the different levels. Visually comparing clustering results based on neighboring levels is simplified by the locally restricted change (split or merge). The bottleneck of AHC in terms of time complexity is the computation of the distance matrix which often requires a vast number of Euclidean distance tests.

5.1.1 Number of Clusters

Salvador and Chan propose a method for automatically computing the number of clusters in hierarchical clustering algorithms [35]. Their *L-method* is based on detecting the *knee* or *elbow* in a graph that opposes the number of clusters and a cluster evaluation metric (Fig. 2). Using the knee, which is defined as the point of maximum curvature of the graph, is well-known. The L-method detects it by finding the two lines that best fit the evaluation graph and then, uses the number of clusters that is closest to their point of intersection. It is obvious that the location of the knee depends of the shape of the graph which again depends on the number of tested cluster numbers. Hence, using a full evaluation graph, which ranges from two clusters to the number of data elements, is recommended. Starting with the full graph, the L-method is carried out iteratively on a decreasing focus region until the current knee location is equal to or larger than the previous location. What is left is the definition of an evaluation measure for constructing the graph. We follow the recommendations of Salvador and Chan who suggest to use the metric employed by the clustering algorithm. Hence, we apply the proximity measure used by the different link versions. Furthermore, they

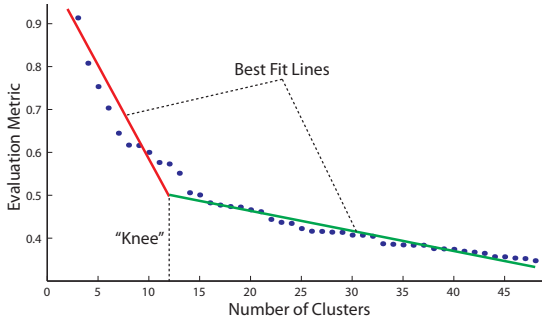


Fig. 2. L-method for automatically computing a reliable number of clusters.

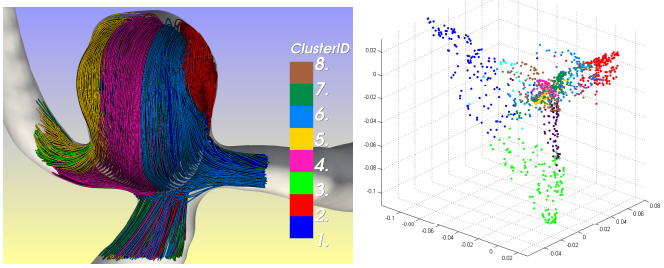
suggest to base the evaluation not on the entire dataset but only on the two clusters that are involved in the current merge step.

5.2 Spectral Clustering

Our explanation of Spectral Clustering (SC) is based on [36] and [37]. SC has become popular in recent years since it often outperforms traditional clustering algorithms such as k-means. One of the key advantages of SC is that it can handle arbitrary cluster shapes. It is called *spectral* because the spectrum of a matrix, i.e. its eigenvalues, plays a central role. The main idea of SC is to map the original data to a new space (*spectral embedding*) where each data entity, e.g., each streamline, is represented as a point (Fig. 3). Key features of the mapping are the preservation of local distance relations between nearby data entities and the enhancement of the data's cluster properties. Enhancement here refers to an improved cluster separability making the subsequent application of standard clustering algorithms feasible. In the following, we briefly overview our implementation of SC and then, we explain some algorithmic details as well as modifications and extensions to the algorithm. We use the terms distance and difference interchangeably, since difference is often expressed by distance. Streamlines serve as our working example.

5.2.1 Graph Partitioning Problem

We have implemented SC as a graph partitioning problem [36] and modified it according to [38]. Streamlines may be represented by a weighted, fully-connected, undirected graph. The nodes of the graph are the streamlines and the edge weights are computed according to a symmetric difference measure. In order for the next steps to work, the edge weights must be transformed from difference to affinity such that similar streamlines have a high and dissimilar a low pairwise affinity. Then, the graph shall be partitioned into two subgraphs. Shi and Malik propose to use a *normalized cut* which minimizes the sum of weights of the edges that need to be removed (cut) and at the same time balances the sum of edge weights of the partitions. Unfortunately, solving this problem is NP hard. However, they show that a relaxed version of this problem may be solved by spectral graph partitioning using *graph Laplacians*.



(a) (b)

Fig. 3. *(a)* Spectral Clustering of streamlines in a basilar tip aneurysm. *(b)* Spectral embedding of the lines. The first three largest eigenvectors are shown.

Let us treat the dataset S with n streamlines as a graph and define a number of clusters k. **(1)** In a first step, the $n \times n$ distance matrix $\mathbf{M}$ is constructed by defining a symmetric distance measure, e.g., according to Equation 2, and applying it in a pairwise fashion to the streamlines in S. **(2)** Based on $\mathbf{M}$, the $n \times n$ weighted adjacency matrix of the graph is constructed by applying a function f to the entries of $\mathbf{M}$ that gives high values in case of small differences and converges to 0 for high differences. The resulting matrix $\mathbf{W}$ is referred to as *affinity matrix*. As a function, the Gaussian similarity function is used:

$$f(m_{ij}) = f(m_{ji}) = \exp(-(m_{ij})^2/(2\sigma^2)) \tag{3}$$

The parameter σ controls the width of the function thereby steering how rapidly the affinity falls off and is mostly user-defined. **(3)** Next, a $n \times n$ diagonal degree matrix $\mathbf{D}$ is constructed with each diagonal entry d_{ii} being the degree of the node that represents streamline i in the graph. The degree is simply computed as the sum of weights of the edges incident to the node. **(4)** Now, the *normalized graph Laplacian* $\mathbf{L}$ is computed (see [37] for other variants of graph Laplacians):

$$L = I - D^{-1}W \tag{4}$$

with $\mathbf{I}$ being the identity matrix. **(5)** Then, the eigenvectors and eigenvalues of $\mathbf{L}$ are computed by its eigendecomposition. The eigenvectors corresponding to the smallest k eigenvalues are then used for clustering. **(6)** Let $\mathbf{U}$ be the $n \times k$ matrix that contains the k eigenvectors as columns. Each row i of $\mathbf{U}$ then represents the coordinates of a point that corresponds to streamline i in the new space spanned by the eigenvectors. The process of mapping the streamline to a point in $\mathbb{R}^k$ is referred to as spectral embedding (Fig. 3(b)). It has been discussed in [37] that this mapping is useful due to properties of graph Laplacians which induce an enhancement of the clustering properties of the data. **(7)** In the new spectral embedding space, clusters can be trivially detected, e.g., by applying k-means.

5.2.2 Local Scaling and Number of Clusters

The discussed algorithm has two user-defined parameters: σ (Eq. 3) and the number of clusters k. For their automatic computation, we adopt and modify the techniques presented by Zelnik-Manor and Perona [38]. They argue in favor of a local determination of σ instead of computing a global value.

The reason is that global values only work well if all clusters are of the same density. Since we cannot guarantee this for the streamlines, we adopt their *local scaling*. Here, a local σ_i is computed for each streamline i based on the difference between i and its N'th neighbor. The Gaussian similarity function from Equation 3 then changes to:

$$f(m_{ij}) = f(m_{ji}) = \exp(-(m_{ij})^2/(\sigma_i\sigma_j)) \quad (5)$$

A value of $N = 7$ has been reported by [38] to give good clustering results. However, our initial experiments indicated that in our case N must be set individually for each dataset. When experimenting with very fine samplings of the ostium surface for streamline generation, we observed a too coarse or too detailed clustering. With an increasing density of streamlines their local neighborhood contains an increasing number of very similar streamlines, i.e., with short streamline-to-streamline distance. According to Equation 5, this leads to very small values in the denominator which in turn results in affinities close to zero. However, the number of neighbors of a data entity with an affinity significantly larger than zero should not be "too small" for SC to work properly [37]. In tests based on ten datasets, we identified setting N to 5% of the streamline count as giving satisfactory and stable results with regard to an increasing ostium sampling.

Zelnik-Manor and Perona suggest an approach for automatically computing the number of clusters k. Instead of specifying k, the user is asked to provide a range of possible values for k. The algorithm then iterates over the range and determines the optimal value. The optimization is based on finding and grading the optimal rotation between the set of the first k largest eigenvectors of $\mathbf{L}$ and the canonical coordinate system. In [38], another graph Laplacian than the variant shown in Equation 4 is used [39]. However, we employ this variant since it is less likely to produce undesired artifacts [37]. Since finding the optimal rotation involves the largest eigenvectors of $\mathbf{L}$, we need to change its definition (Eq. 4) to:

$$L = D^{-1}W \quad (6)$$

5.2.3 Summary and Comparison

In summary, SC strives for an optimal partitioning from a global perspective while hierarchical clustering aims at making good local decisions. SC can handle arbitrary cluster shapes. It is biased towards clusters of similar size due to the balancing of edge weights in the graph cutting. On the other hand, this property makes it robust against outliers which was acknowledged in the context of fiber tract length [29]. Our implementation of SC is parameter-free except for the range of possible numbers of clusters. Since the algorithm computes all clusterings within this range during optimization, the user could also interactively browse the suboptimal results. However, visually detecting changes is harder than in hierarchical clustering since they may not be restricted to a local region. An advantage of SC using local scaling over hierarchical clustering is its consideration of local changes in cluster density. This may be particularly useful if streamlines are seeded with a non-uniform density, e.g., a higher density near to the aneurysm wall. As for hierarchical clustering, the bottleneck in terms of time complexity is the computation of the distance matrix. Table 1 compares both clustering algorithms with respect to important capabilities.

Capability	Spectral Clustering	Agglomerative Hierarchical Clustering			
		Single	Complete	Average	Ward
Shape	+	+	−	−	−
Size	*o*	+	−	*o*	*o*
Outlier	+	−	*o*	*o*	+

TABLE 1
Capabilities of Agglomerative Hierarchical Clustering and Spectral Clustering to handle arbitrarily-shaped clusters, clusters of significantly different size, and outliers.

6 Evaluation of Streamline Clustering Techniques

We evaluate Spectral Clustering (SC) and four versions of Agglomerative Hierarchical Clustering (AHC) based on ten datasets. Important dataset characteristics, the respective automatically computed number of clusters and the time for computing this number are listed in Table 2. Note that all timings include the clustering itself since it is part of determining the optimal number of clusters.

AHC and SC including the computation of inter-streamline distances are implemented in MATLAB (MathWorks, Natick, MA, U.S.). Source code of the L-method is provided by Athanassios Zagouras as part of MATLAB Central's file exchange [40]. Source code for the local scaling and the automatic determination of the cluster number by means of eigenvector rotation is provided by Zelnik-Manor and Perona [41]. The computation of distances between streamlines has been parallelized to work on multi-core architectures by means of MATLAB's Parallel Computing Toolbox. The entire code is exported as a shared library which may then be accessed from standard C++ code.

Dataset Characteristics: Two different types of saccular aneurysms are represented by the datasets: side-wall and basilar tip aneurysms. The datasets of the Virtual Intracranial Stenting Challenges (VISC) in 2009 and 2010 comprise virtually stented aneurysms (NF = Neuroform stent, SK = SILK stent). The aneurysms in the remaining datasets have not been stented virtually. The 4th column of the table lists the number of generated streamlines and the 5th column lists the average number of streamline points. Although the same ostium mesh has been used for the untreated and the stented case of VISC 2009, their line numbers differ slightly. This is due to the integration during streamline generation, which failed to start at a few different vertices. For the datasets of VISC 2009 and VISC 2010, we employed a very fine-grained sampling of the ostium leading to a high number of lines. However, the streamlines of VISC10_NF_L and VISC10_SK_L have been generated using the initial sampling since the stent positioning therein has proven to be less beneficial for the patient [42].

No.	Dataset	Dataset Characteristics			Number of Clusters (k)					Time for Computing Number of Clusters [s]				
		Aneurysm	# Lines	# Points (ø)	Single	Complete	Average	Ward	sc	Single	Complete	Average	Ward	SC
1	MD	side-wall	1816	190	3	10	6	10	9	285	260	264	247	14
2	HB	basilar tip	1178	348	39	5	10	7	8	121	114	120	116	7
3	PS	basilar tip	1158	310	9	5	9	9	11	121	112	131	110	6
4	VISC09	side-wall	2254	505	8	6	10	9	6	457	371	390	358	20
5	VISC09_SK		2207	249	49	8	8	3	7	344	335	354	336	20
6	VISC10	basilar tip	2929	265	15	10	6	5	8	606	609	615	620	41
7	VISC10_NF_R		2923	275	3	8	7	7	8	598	600	629	615	40
8	VISC10_NF_L		1153	283	14	6	7	9	15	116	109	119	108	7
9	VISC10_SK_R		2891	234	9	6	4	3	12	719	582	613	588	40
10	VISC10_SK_L		1138	212	3	9	5	6	9	123	108	110	104	10

TABLE 2
Datasets used in evaluating Agglomerative Hierarchical Clustering (four versions) and Spectral Clustering (SC). Dataset characteristics, the automatically computed number of clusters, and the time for computing this number are listed. Cell colors indicate whether k is unacceptable (red), inadequate (yellow), or appropriate (green).

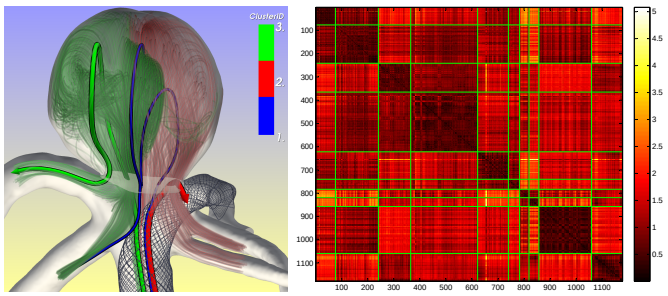
6.1 Number of Clusters

In this section, we investigate which clustering technique, in conjunction with its method for automatically determining a number of clusters k, consistently returns a reliable k. The columns 6 to 10 of Table 2 show the values of k that have been automatically computed using a given search range. All AHC versions employed a range of $k = [2, \#lines]$. In SC, we empirically determined a range of $k = [4, 20]$ to be sufficient for detecting the relevant flow structures in the ten datasets. The numbers of clusters of each dataset are fairly different across the algorithms. Consistencies rarely occur and show no pattern. Strong deviations are observed mostly for AHC with single link. In order to determine the algorithm that computes the most reliable k for each dataset, we carefully inspected the clustering results. A reliable k should lead to clusters characterized by:

- low intra-cluster streamline variance (*high cohesion*), and
- high inter-cluster streamline variance (*high separation*).

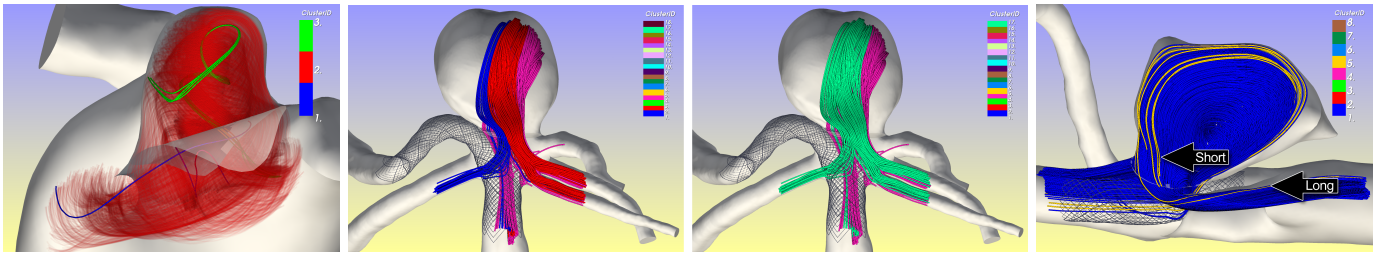
To fulfill these requirements, the number of clusters must not be too small leading to a low cohesion due to the aggregation of significantly different streamlines. Further, it should not be too high resulting in a low separation due to similar streamlines in different clusters. We put special emphasis on a high cohesion since in a flow summary visualization based on one representative per cluster missing flow features are less tolerable than overrepresented ones. In order assess the reliability of a number of clusters, we concurrently employed three different approaches. All streamlines were colored according to cluster ID and the possibility to hide and show individual clusters was added (Fig. 3(a)). This facilitates a coarse assessment of cohesion and separation. Further, the visualization could be restricted to streamlines having a distance larger than an interactively adjustable threshold to their cluster representative (Fig. 4(a)). If many of such streamlines together represent an additional flow structure, the cohesion is poor and k is too small. For instance, the transparent green lines in Figure 4(a) indicate a swirl which is missing from the visualization via representatives. In order to further support the assessment of cohesion and separation, the distance matrix $\mathbf{M}$ is sorted according to cluster ID and displayed as a colored image (Fig. 4(b)). The cluster borders are indicated by overlaid lines. For well-separated clusters with a good cohesion, the reordered matrix should have a block diagonal structure. In other words, distances inside clusters should be small (dark colors) and distances outside clusters should be high (bright colors). The matrix view indicates whether a value for k is too small (bright colors inside diagonal blocks) or too high (dark colors outside diagonal blocks as for the two upper leftmost clusters in Figure 4(b)).

Besides searching for a reliable value of k, we reviewed all clustering results with respect to the issues which have been discussed in the context of AHC (Sec. 5.1). In single link clustering, chaining was observed for nine datasets. A single cluster contains almost every streamline while the remaining clusters mostly comprise only a single line (Fig. 5(a)). This effect was tackled by Yan et al. through a subsequent top-down balancing of the cluster tree [10]. The trend to break large clusters was observed in complete link clustering for three datasets. Here, two similar parts of a large cluster were not merged due to their high maximum distance. Instead, one part is merged with a smaller, less similar, but spatially closer cluster (Fig. 5(b)). Despite the adapted streamline similarity measure (Sec. 4.2), average link's sensitivity to outliers lead to very small-sized clusters in three datasets (Fig. 5(c)). The effect was also observed for complete link in one dataset. It is caused by short outliers, which are not similar enough to neighboring streamlines and are hence grouped in a separate



(a) (b)

Fig. 4. *(a)* Streamlines deviating strongly from their cluster representative (tubes). *(b)* Distance matrix reordered according to cluster ID. Lines indicate cluster borders.



(a) (b) (c)

Fig. 5. Clustering issues. *(a) Chaining* in Agglomerative Hierarchical Clustering (AHC) with single link. A large, red cluster is generated containing almost every streamline. *(b)* AHC with complete link tends to break large clusters. Instead of merging the red and magenta streamlines (left), the red lines are merged with the small blue cluster in the next step (right). *(c)* AHC with average link is still sensitive to outlier streamlines (short, yellow lines) if they deviate too much from neighboring lines (blue). Outliers are grouped separately or with a few, long, "correct" lines.

cluster. Although only a subset of outliers is separated this way and the generated cluster may as well contain long, "correct" streamlines, this may be considered a feature of the algorithm. It could be exploited in an outlier removal preprocessing step. Note that small-sized, outlier corrupted clusters occurred in each dataset with average link based on the original MCPD measure.

In order to identify a reliable number of clusters k for each dataset, we assigned the computed numbers to three categories. They were rated as unacceptable if clustering issues were detected, as inadequate if k was too small or too high, and as appropriate. The cells of Table 2 are colored accordingly in red, yellow, and green. The eigenvector rotation associated with SC delivers a reliable number of clusters k_{sc} in almost every case. Furthermore, it shows the best computational performance as can be inferred from columns 11 to 15. However, the performance of the L-method may be improved by cutting off unlikely high numbers of clusters from the full evaluation graph used for determining k [35].

6.2 Quality of Clustering Results

Having identified a method, which consistently returns a reliable number of clusters k_{sc}, we seek the clustering algorithm which produces the best results based on k_{sc}. Different measures of goodness have been proposed in clustering literature. In the absence of a ground truth, unsupervised measures of cluster validity are appropriate [32]. They are also called *internal indices* since they are purely based on information present in the data. We employed three internal indices which measure different aspects of the data (see [33] for a survey and implementation details):

- *Silhouette Width*: Non-linear combination measure of cluster cohesion and separation. Values are in the range $[-1,+1]$ and should be maximized.
- *Connectivity*: Local measure reflecting the degree to which the L closest streamlines are placed in the same cluster. Values are in the range $[0,+\infty]$ and should be minimized. In our computations, we set $L = 20$.
- *Hubert's* Γ *Statistic*: Measure of correlation between the distance matrix $\mathbf{M}$ and an idealized distance matrix (distance is 0 for streamlines in the same cluster and 1, otherwise). Values of the normalized statistic are in the range $[-1,+1]$ and should be maximized.

After each AHC version was forced to generate k_{sc} clusters, the internal indices were computed. The results are presented in Figure 6. Bars with no filling correspond to clustering results rated as unreliable due to artifacts caused by outliers. The silhouette width of single link is unsurprisingly poor, due to the chaining effect. Complete link performs better having an average width of 0.28. Average link, Ward's method and SC perform equally well and exhibit the highest silhouette widths. A drawback of the silhouette is that it favors algorithms creating globular clusters [33]. Hence, algorithms which correctly identify elongated or concave clusters, e.g., single link and SC, may be assigned a lower silhouette than failing algorithms. Since in clustering streamlines only distances between data entities but not the cluster structure itself is known, the assumption of globular clusters may be invalid and the silhouette width must be employed carefully. For fiber tracts, the non-globular nature of clusters has already been acknowledged [27].

Single link clustering by far achieves the best connectivity values. This is due to its proximity measure which strives for a merge with the nearest neighbor. This bias has already been acknowledged in [33]. The second and third best connectivity values are achieved variably by average link and Ward's method. Complete link exhibits the highest values of all AHC versions since its proximity measure is based on the furthest neighbor. The connectivity of SC also shows high values. The fact that single link, average link and Ward's method lead to better connectivity values is not justified by a better clustering result but by the way how the algorithms work and how connectivity is computed. The computation starts with a connectivity of zero. It adds the highest penalty value if the nearest neighbor is not included in the same cluster. This rarely occurs in AHC since each version starts by aggregating the nearest singleton clusters (Fig. 7). SC aims at a global optimization and occasionally adds the nearest neighbor to another cluster if this is beneficial for the final result. Due to the bias of connectivity towards the AHC approaches, its usefulness in assessing SC is questionable. Also for single link it may not be expressive since it does not reflect chaining

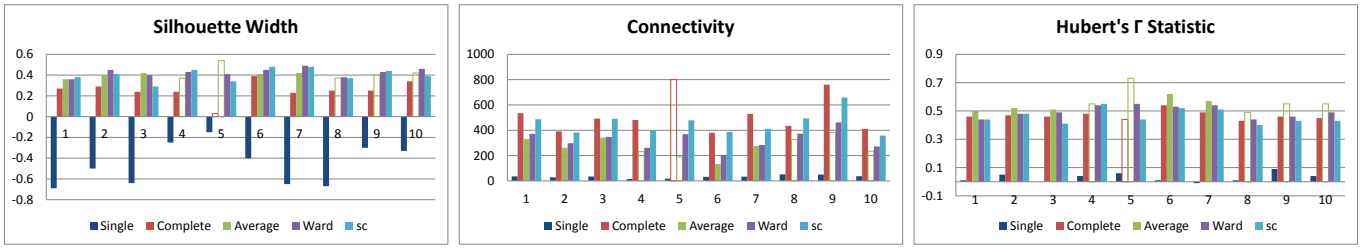


Fig. 6. Internal cluster indices of the datasets in Table 2. Agglomerative Hierarchical Clustering (four versions) and Spectral Clustering (SC) are compared. Clustering was carried out using the number of clusters suggested by SC.

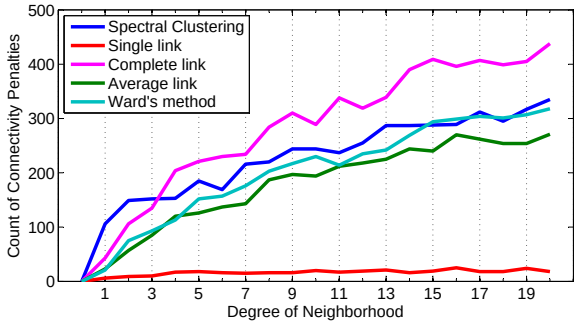


Fig. 7. Count of connectivity penalties of dataset 1 in Table 2. If the ith nearest neighbor of streamline s is not in the same cluster as s, a penalty is assigned ($i \in [1,20]$).

which occurred in all clustering results based on k_{sc}.

Hubert's Γ Statistic shows poor results for single link due to the chaining effect. In the one large cluster, very dissimilar streamlines are grouped together leading to very poor correlation values. Complete link, Ward's method and SC exhibit similar results on average. The highest values are obtained for average link by a rather narrow margin. Again SC suffers from not promoting the assignment of nearest neighboring streamlines to the same cluster in the beginning. Small distances between streamlines in different clusters are hardly penalized by Hubert's Γ Statistic. Hence, this measure also favors algorithms creating globular clusters.

7 Conclusion

We evaluated *Spectral Clustering* and *Agglomerative Hierarchical Clustering* with single link, complete link, average link, and Ward's method in the context of blood flow clustering. The evaluation of each technique included a method for automatically determining a reliable number of clusters. It was based on streamlines generated for ten datasets conveying the flow patterns in five different intracranial aneurysms.

The eigenvector rotation associated with Spectral Clustering delivered a reliable number of clusters k_{sc} in 9 of 10 datasets. The L-method associated with Agglomerative Hierarchical Clustering and Ward's method placed a distant second with 5 of 10 datasets. Furthermore, the eigenvector rotation performed a factor of $16-17$ faster than the original implementation of the L-method. However, the performance of the L-method may be improved by cutting off unlikely high numbers of clusters from the full evaluation graph used for determining k [35]. Note that the timings in the last column of Table 2 also refer to the entire clustering process, since the determination of a reliable k includes the clustering itself.

Having identified a reliable number of clusters for each dataset, all clustering techniques were reapplied but this time based on k_sc. In order to assess the goodness of each clustering result, three internal cluster indices were computed: silhouette with, connectivity, and Hubert's Γ statistic. External indices could not be applied since no ground truth exists for clustering blood flow, neither with respect to the number of clusters nor regarding the grouping of similar lines.

In summary, single link is not suitable for clustering blood flow because of the chaining effect. Complete link generates better clusters but its tends to break large clusters. Further, its connectivity is rather poor and its silhouette width indicates a worse cluster cohesion and/or separation as compared to the remaining algorithms. Average link shows good results for all internal indices despite being sensitive to outliers. This sensitivity could be exploited in a preprocessing step including outlier detection and removal. Ward's method and Spectral Clustering perform equally well with respect to the silhouette width and Hubert's Γ Statistic and they are not sensitive to outlier streamlines. Together with average link after outlier removal, they can be recommended for clustering blood flow. With regard to a fully automatic clustering however, average link and Ward's method should be coupled either with a different method for automatically determining a reliable k or the L-method should be tested with other evaluation metrics.

Further studies may investigate the overlap of the different clustering results. Initial comparisons based on the Rand Index (*RDI*) and the Adjusted Rand Index (*ARDI*) [43] revealed on average a considerable overlap but also differences which are worth investigating: average link vs. Ward's method: $RDI = 0.89$ and $ARDI = 0.60$, average link vs. SC: $RDI = 0.85$ and $ARDI = 0.48$, and Ward's method vs. spectral clustering: $RDI = 0.91$ and $ARDI = 0.59$. Values of the RDI and ARDI are in the range $[0,1]$ and should be maximized.

REFERENCES

[1] R. D. Brown, J. Huston, R. Hornung, T. Foroud, D. F. Kallmes, D. Kleindorfer, I. Meissner, D. Woo, L. Sauerbeck, and J. Broderick, "Screening for Brain Aneurysm in the Familial Intracranial Aneurysm Study: Frequency and Predictors of Lesion Detection," *Neurosurgery: Pediatrics*, vol. 108, no. 6, pp. 1132–1138, 2008.

[2] L. Augsburger, P. Reymond, E. Fonck, Z. Kulcsár, M. Farhat, M. Ohta, N. Stergiopulos, and D. Rüfenacht, "Methodologies to Assess Blood Flow in Cerebral Aneurysms: Current State of Research and Perspectives," *Neuroradiology*, vol. 36, no. 5, pp. 270–277, 2009.

[3] J. R. Cebral, M. A. Castro, J. E. Burgess, R. S. Pergolizzi, M. J. Sheridan, and C. M. Putman, "Characterization of Cerebral Aneurysms for Assessing Risk of Rupture by Using Patient-Specific Computational Hemodynamics Models," *American Journal of Neuroradiology*, vol. 26, no. 10, pp. 2550–2559, 2005.

[4] J. R. Cebral, F. Mut, J. Weir, and C. M. Putman, "Association of Hemodynamic Characteristics and Cerebral Aneurysm Rupture," *American Journal of Neuroradiology*, vol. 32, no. 2, pp. 264–270, 2011.

[5] S. Appanaboyina, F. Mut, R. Löhner, C. Putman, and J. Cebral, "Simulation of intracranial aneurysm stenting: Techniques and challenges," *Computer Methods in Applied Mechanics and Engineering*, vol. 198, no. 45-46, pp. 3567 – 3582, 2009.

[6] M. Kim, E. I. Levy, H. Meng, and L. N. Hopkins, "Quantification of hemodynamic changes induced by virtual placement of multiple stents across a wide-necked basilar trunk aneurysm." *Neurosurgery*, vol. 61, no. 6, pp. 1305–1312; discussion 1312–1313, 2007.

[7] I. Larrabide, M. L. Aguilar, H. G. Morales, A. J. Geers, Z. Kulcsár, D. Rüfenacht, and A. F. Frangi, "Intra-Aneurysmal Pressure and Flow Changes Induced by Flow Diverters: Relation to Aneurysm Size and Shape," *AJNR Am J Neuroradiol*, p. [Epub ahead of print], 2012. [Online]. Available: http://dx.doi.org/10.3174/ajnr.A3288

[8] T. McLoughlin, M. W. Jones, R. S. Laramee, R. Malki, I. Masters, and C. D. Hansen, "Similarity measures for enhancing interactive streamline seeding," *IEEE Transactions on Visualization and Computer Graphics*, p. [Epub ahead of print], 2012. [Online]. Available: http://dx.doi.org/10.1109/TVCG.2012.150

[9] C. Rössl and H. Theisel, "Streamline Embedding for 3D Vector Field Exploration," *IEEE Trans Vis Comput Graph*, vol. 18, no. 3, pp. 407–420, 2012.

[10] H. Yu, C. Wang, C.-K. Shene, and J. H. Chen, "Hierarchical streamline bundles." *IEEE Trans Vis Comput Graph*, vol. 18, no. 8, pp. 1353–1367, 2012.

[11] B. Moberts, A. Vilanova, and J. van Wijk, "Evaluation of Fiber Clustering Methods for Diffusion Tensor Imaging," in *IEEE Visualization*, 2005, pp. 65 – 72.

[12] L. J. O'Donnell and C.-F. Westin, "Automatic tractography segmentation using a high-dimensional white matter atlas," *IEEE Transactions on Medical Imaging*, vol. 26, no. 11, pp. 1562–1575, 2007.

[13] A. R. Mantha, G. Benndorf, A. Hernandez, and R. W. Metcalfe, "Stability of pulsatile blood flow at the ostium of cerebral aneurysms." *J Biomech*, vol. 42, no. 8, pp. 1081–1087, May 2009.

[14] R. Gasteiger, M. Neugebauer, O. Beuing, and B. Preim, "The FLOWLENS: A Focus-and-Context Visualization Approach for Exploration of Blood Flow in Cerebral Aneurysms," *IEEE Transactions on Visualization and Computer Graphics (TVCG)*, vol. 17, no. 12, pp. 2183–2192, 2011.

[15] J. Schöberl, "NETGEN An Advancing Front 2D/3D-Mesh Generator Based on Abstract Rules," *Computing and Visualization in Science*, vol. 1, pp. 41–52, 1997.

[16] M. Neugebauer, G. Janiga, O. Beuing, M. Skalej, and B. Preim, "Anatomy-Guided Multi-Level Exploration of Blood Flow in Cerebral Aneurysms," *Computer Graphics Forum (EuroVis)*, vol. 30(3), pp. 1041–1050, 2011.

[17] J. R. Cebral, F. Mut, J. Weir, and C. M. Putman, "Quantitative Characterization of the Hemodynamic Environment in Ruptured and Unruptured Brain Aneurysms," *American Journal of Neuroradiology*, vol. 32, no. 1, pp. 145–151, 2011.

[18] F. H. Post, B. Vrolijk, H. Hauser, R. S. Laramee, and H. Doleisch, "The State of the Art in Flow Visualisation: Feature Extraction and Tracking," *Comput. Graph. Forum*, vol. 22, no. 4, pp. 775–792, 2003.

[19] T. Salzbrunn, H. Jänicke, T. Wischgoll, and G. Scheuermann, "The state of the art in flow visualization: Partition-based techniques," in *SimVis*, 2008, pp. 75–92.

[20] T. Salzbrunn and G. Scheuermann, "Streamline predicates," *Visualization and Computer Graphics, IEEE Transactions on*, vol. 12, no. 6, pp. 1601 –1612, 2006.

[21] S. Born, M. Pfeifle, M. Markl, and G. Scheuermann, "Visual 4d mri blood flow analysis with line predicates," in *Pacific Vis*, 2012, pp. 105–112.

[22] K. Shi, H. Theisel, H. Hauser, T. Weinkauf, K. Matković, H.-C. Hege, and H.-P. Seidel, "Path line attributes - an information visualization approach to analyzing the dynamic behavior of 3D time-dependent flow fields," in *Topology-Based Methods in Visualization II*, ser. Mathematics and Visualization. Springer, 2009, pp. 75–88.

[23] A. Pobitzer, A. Lež, K. Matković, and H. Hauser, "A Statistics-based Dimension Reduction of the Space of Path Line Attributes for Interactive Visual Flow Analysis," in *Pacific Vis*, 2012, pp. 113–120.

[24] R. van Pelt, J. Olivan Bescos, M. Breeuwer, R. Clough, M. Groller, B. ter Haar Romeny, and A. Vilanova, "Interactive virtual probing of 4d mri blood-flow," *Visualization and Computer Graphics, IEEE Transactions on*, vol. 17, no. 12, pp. 2153–2162, 2011.

[25] K. Bidmon, S. Grottel, F. Bös, J. Pleiss, and T. Ertl, "Visual Abstractions of Solvent Pathlines near Protein Cavities," *Computer Graphics Forum*, vol. 27, no. 3, pp. 935–942, 2008.

[26] C.-K. Chen, S. Yan, H. Yu, N. Max, and K.-L. Ma, "An illustrative visualization framework for 3d vector fields." *Comput. Graph. Forum*, vol. 30, no. 7, pp. 1941–1951, 2011.

[27] J. Klein, P. Bittihn, P. Ledochowitsch, H. K. Hahn, O. Konrad, J. Rexilius, and H.-O. Peitgen, "Grid-based Spectral Fiber Clustering," in *SPIE Medical Imaging: Visualization and Image-Guided Procedures*, vol. 6509, 2007, pp. 65 091E–65 091E–10. [Online]. Available: + http://dx.doi.org/10.1117/12.706242

[28] L. O'Donnell, A. J. Golby, and C.-F. Westin, "Tract-based morphometry for white matter group analysis," *NeuroImage*, vol. 45, pp. 832–844, 2009.

[29] S. Zhang, S. Correia, and D. Laidlaw, "Identifying white-matter fiber bundles in dti data using an automated proximity-based fiber-clustering method," *IEEE Transactions on Visualization and Computer Graphics*, vol. 14, no. 5, pp. 1044 –1053, 2008.

[30] M. Shojima, M. Oshima, K. Takagi, R. Torii, M. Hayakawa, K. Katada, A. Morita, and T. Kirino, "Magnitude and role of wall shear stress on cerebral aneurysm: Computational fluid dynamic study of 20 middle cerebral artery aneurysms," *Stroke*, vol. 35, no. 11, pp. 2500–2505, 2004.

[31] I. Corouge, S. Gouttard, and G. Gerig, "Towards a shape model of white matter fiber bundles using diffusion tensor mri," in *ISBI*, 2004, pp. 344–347.

[32] P.-N. Tan, M. Steinbach, and V. Kumar, *Introduction to Data Mining*. Addison Wesley, 2005.

[33] J. Handl, J. Knowles, and D. B. Kell, "Computational cluster validation in post-genomic data analysis." *Bioinformatics*, vol. 21, no. 15, pp. 3201–3212, Aug 2005.

[34] L. Ferreira and D. B. Hitchcock, "A comparison of hierarchical methods for clustering functional data," *Communications in Statistics - Simulation and Computation*, vol. 38, no. 9, pp. 1925–1949, 2009.

[35] S. Salvador and P. Chan, "Determining the Number of Clusters/Segments in Hierarchical Clustering/Segmentation Algorithms," in *Tools with Artificial Intelligence, 2004. ICTAI 2004. 16th IEEE International Conference on*, 2004, pp. 576 – 584.

[36] J. Shi and J. Malik, "Normalized cuts and image segmentation," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 22, no. 8, pp. 888 –905, aug 2000.

[37] U. von Luxburg, "A tutorial on spectral clustering," *Statistics and Computing*, vol. 17, no. 4, pp. 395–416, 2007.

[38] L. Zelnik-Manor and P. Perona, "Self-tuning Spectral Clustering," in *Advances in Neural Information Processing Systems 17*. MIT Press, 2004, pp. 1601–1608.

[39] A. Y. Ng, M. I. Jordan, and Y. Weiss, "On spectral clustering: Analysis and an algorithm," in *Advances in Neural Information Processing Systems (NIPS)*, 2001, pp. 849–856.

[40] "A. Zagouras. L-method," www.mathworks.com/matlabcentral/fileexchange/37295-l-method/content/Lmethod.m.

[41] "L. Zelnik-Manor. Self-Tuning Spectral Clustering - MATLAB sources," www.vision.caltech.edu/lihi/Demos/SelfTuningClustering.html.

[42] G. Janiga, C. Rössl, M. Skalej, and D. Thvenin, "Realistic virtual intracranial stenting and computational fluid dynamics for treatment analysis," *Journal of Biomechanics*, no. 0, p. [Epub ahead of print], 2012. [Online]. Available: www.sciencedirect.com/science/article/pii/S0021929012005234

[43] L. Hubert and P. Arabie, "Comparing partitions," *Journal of classification*, vol. 2, no. 1, pp. 193–218, 1985.