Vision, Modeling, and Visualization (2018)
F. Beck, C. Dachsbacher, and F. Sadlo (Eds.)

Interactive Visual Exploration of Line Clusters

Mathias Kanzler and Rüdiger Westermann

Technical University of Munich (TUM), Germany

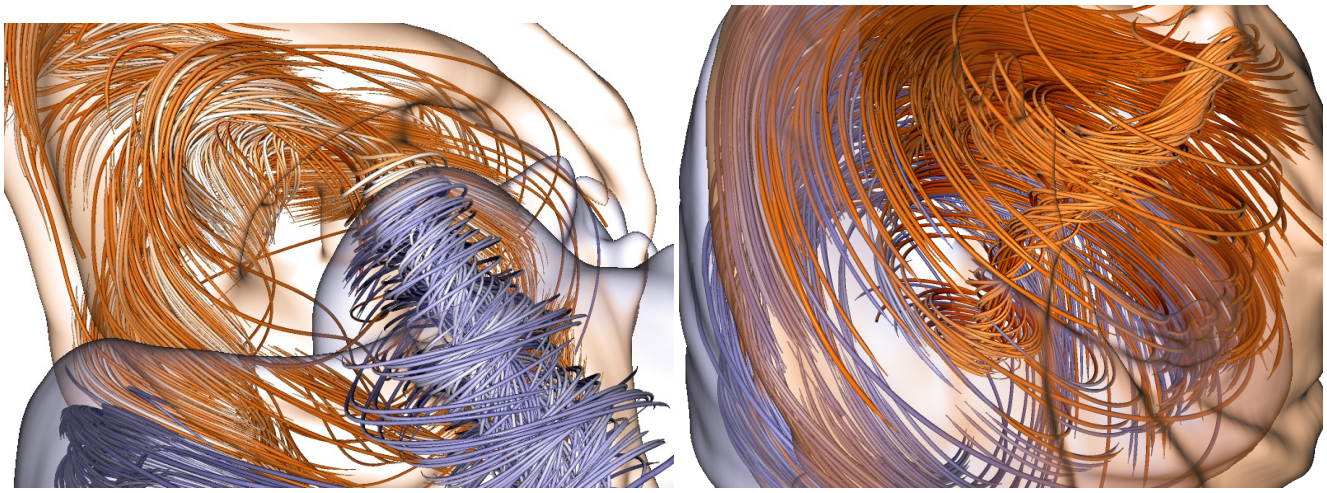

Figure 1: *Clusters of lines are analyzed by visualizing cluster-specific contextual information in the form of cluster hulls (colored surfaces), and using a measure of local line consistency within a cluster to adapt the line density automatically so that important structures are conveyed. Via brushing the user can interactively select where automated density control is applied.*

Abstract

We propose a visualization approach to interactively explore the structure of clusters of lines in 3D space. We introduce cluster consistency fields to indicate the local consistency of the lines in a cluster depending on line density and dispersion of line directions. Via brushing the user can select a focus region where lines are shown, and the consistency fields are used to automatically control the density of displayed lines according to information content. The brush is automatically continued along the gradient of the consistency field towards high information regions, or along a derived mean direction field to reveal major pathways. For a given line clustering, visualizations of cluster hulls are added to preserve context information.

1. Introduction

Clustering of lines is used in many applications [MVvW05, AT06, AMP10, YWSC12] to condense large line sets to few representative lines that expose the major geometric trends in the initial set. Here, we refer to a line as a sequence of vertices where every pair of consecutive vertices is connected by a straight line segment. While the representative lines serve as a basis for the application-specific analysis process, the distribution and spatial coverage of the lines per cluster, and the variation of these characteristics among clusters is usually not examined any further. Instead, it is quietly assumed that the lines in one cluster follow the representative line closely.

For large line sets and a reasonable number of clusters, however, the lines in one cluster usually show significant local deviations from the representative, requiring indicators of where deviations occur and how these deviations reflect in the lines' geometries. Computing a large enough set of clusters so that the per-cluster deviations are small, on the other hand, contradicts the requirement to effectively reduce the number of representatives for the upcoming analysis process. Thus, one cannot avoid significant local deviations in general, yet one can try to analyze these deviations in 3D space to better interpret the consistency of grouped elements. Such an analysis is also important to compare different clusterings

DOI: 10.2312/vmv.20181265

to each other, and to eventually shed light on specific properties of the used clustering algorithm and distance metric.

To perform a consistency analysis of a given cluster, the lines belonging to that cluster can be shown in addition to the representative line, either entirely or only to a certain percentage [YWSC12, OLK*14]. While the first approach can quickly introduce occlusions, the second one can withhold important information if the lines are thinned out randomly. Yu et al. [YWSC12] address these shortcomings by showing in addition to the representative line also lines indicating the cluster boundaries. This can reveal the spatial extent of a cluster, but it cannot effectively reveal a cluster's consistency in the interior.

Our contribution: Building upon these previous works, our approach enables an interactive visual exploration of the local consistency of a cluster of lines, in context of the clusters in the current clustering. We introduce a measure of cluster consistency in 3D space, which considers the local density of lines as well as the variance of the local line directions. Boundary surfaces in the scalar-valued consistency field are used to indicate a cluster's shape.

Starting with a context view showing cluster boundaries and representatives, we let the user brush a region where lines should be added. By using a hierarchy of line clusters [KFW16], the line density in the brushed region is controlled to emphasize regions where lines are strongly diverging, at the same time showing only few representative lines in regions where lines behave similar. The object-space cluster consistency is used to determine the line density. To further guide the user towards regions of low consistency, the brush is continued automatically along the gradient of the consistency field. Alternatively, the brush can be continued along the major trends in the line set by following bundles of similar lines.

Our specific contributions are:

- *Cluster consistency fields* as spatial indicators for the line consistency in a cluster.
- The use of cluster consistency fields to reveal the most representative lines in a brushed focus region.
- A method to continue a user-selected brush along major pathways and towards regions of high information content.

We demonstrate our approach by visualizing a number of real-world examples comprised of large line sets. Some of these data sets are shown in Fig. 1, hinting towards the specific visualizations targeted in this work. Equipped with a cluster panel that shows an abstract view of a given line clustering, the user can interactively select clusters based on their consistency, and compare clusters using their hulls and overlaps.

2. Related Work

Our approach uses a hierarchical clustering of a line set. Let us refer to the book by Jain [Jai10] for an exhaustive summary of available clustering algorithms. An overview of similarity measures for lines is given in the comparative study by Zhang et al. [ZHT06]. Oeltze and co-workers [OLK*14] evaluate clustering approaches for streamlines using geometry-based similarity measures. Different clustering approaches and similarity measures for fiber tracts in diffusion tensor imaging data have been evaluated by Moberts et al. [MVvW05].

Yu et al. [YWSC12] use hierarchical clustering to generate a small representative set of streamlines in a flow field. They further select streamlines close to the cluster boundaries to convey the spatial extent of computed clusters. Kanzler et al. [KFW16] use a line cluster hierarchy for view-dependent line density control. Every segment of a line determines a level in the pre-computed line hierarchy depending on a screen-space importance measure, and only if the line is the representative line at this level the segment is drawn. For the construction of a line cluster hierarchy we follow the work by Kanzler et al., which uses a variant of Agglomerative Hierarchical Clustering (AHC) with single linkage and graph matching to construct a fully balanced cluster tree.

Both aforementioned approaches try to find the minimum number of lines that represent the important structures in a given line set. This process can be performed either via importance- or similarity-based criteria in object-space, or in screen-space by selecting the rendered lines dynamically on a frame-to-frame basis. Screen-space approaches determine for each new view the subset of lines to be rendered so that occlusions are reduced and more important lines are favored over less important ones [MCHM10, LMSC11, MWS13]. Indicators for the amount of occlusion in the rendered images are based on the "overdraw", i.e., the number of projected line points per pixel [MCHM10, MWS13], or the maximum projected entropy per pixel [LMSC11]. View-dependent opacity control was introduced by Guenther et al. [GRT13] to fade out those parts of foreground lines that occlude more important ones.

For the selection of representative lines in object-space a number of different criteria have been proposed, for instance, based on the line density [TB96, MHHI98, MTHG03, SHH*07], the line distribution [JL97, LMG06, LHS08, SLCZ09, RPP*09], or the coverage of specific flow or line features [VKP00, YKP05, CCK07, MJL*12]. Behrendt et al. [BBB*18] suggest an explorative approach to select lines based on surface features. The explorative selection of point clouds can be accomplished by encircling a target location as described by Yu et al. [YEII12]. An interactive focus and context approach using a 2D lens to analyze flow structures is described by Gasteiger et al. [GNBP11] and Fuhrmann et al. [FG98]. While Jackson et al. [JCK12] are using a dedicated haptic device to explore flow structures in 3D, our method is designed for standard input devices (mouse and keyboard). Our proposed consistency measure to control the line density is related to the measure of information entropy that was introduced by Xu et al. [XLS10]. They use the local directional variation of a given vector field as entropy measure, and generate an entropy field that is then used to guide the placement of streamlines. We introduce a similar measure using the local directional variations of lines in a cluster, and we further extend this measure to also consider the line density.

3. Method Overview

The input of our method consists of a set of lines, e.g., streamlines, where each line is represented by a sequence of vertices. If no initial clustering is given, we use the mean-of-closest-point dis-

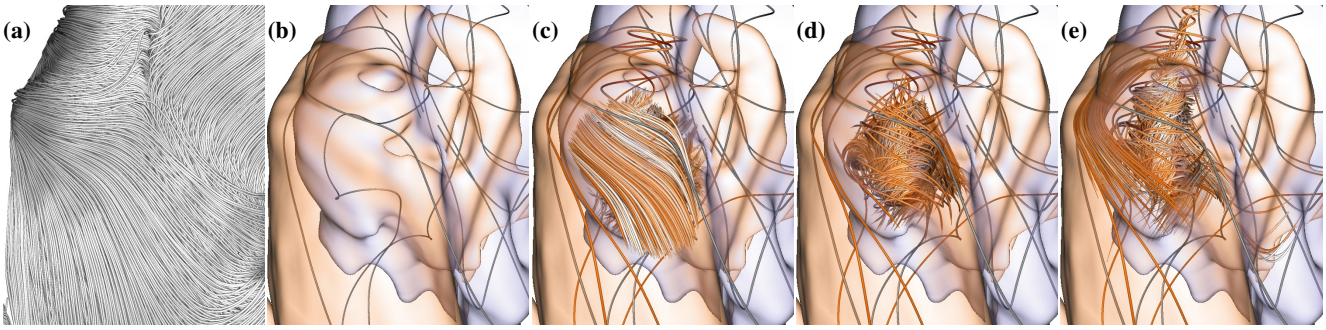

Figure 2: *(a) Initial line set. (b) Representative lines of 8 clusters and hulls of two clusters. (c) All lines going through a brushed focus region. (d) View-dependent line density control using local consistency. (e) Automatic refinement and continuation of focus region towards important structures in the surrounding.*

tance [CGG04] as distance metric and compute a balanced cluster tree using graph matching [KFW16].

Every node in the computed cluster tree represents a cluster of lines that are grouped together based on the used distance metric. For selecting a single representative line at each node, we perform a top-down sweep through the tree as follows: We traverse the tree in breadth-first order, and whenever we encounter a node that has not been visited we determine the representative line, i.e., the line that has the smallest average mean-of-closest-point distance to all lines in the cluster at that node. This line becomes the representative line for every cluster at the nodes along the path from the current node to the leaf node where the line is the only cluster member. In this way we build a hierarchy of representatives which explain as much as possible of the cluster variation at the highest (coarsest) levels. In addition, the hierarchy is nested in that at every inner node one of the child nodes has the same representative than that node. This enables to keep the current representatives and progressively add the new ones when going from one level to the next finer one. Fig. 2 (a) and (b), respectively, show a line set and the 8 representative lines (as well as two cluster hulls) at level three of the computed cluster hierarchy.

For every cluster down to a user-selected level a cluster consistency field (CCF) is pre-computed (c.f. Section 4). If at run-time a CCF is requested that has not been computed, it is generated in turn within less than a second on the GPU. A CCF is similar to a visitation volume [Jon08, BCM*08, BFMW12], which is generated by rasterizing lines into a 3D voxel grid and counting the number of lines going through every voxel. Level sets in this field can then be visualized to show the spatial extend of the lines in one cluster, i.e., the cluster hull (see Fig. 2 (b)), and multiple hulls can be displayed concurrently to enable a comparative cluster analysis. In addition to line density, we also store a measure of the directional variance of the lines going through a voxel.

While cluster hulls provide context information about a cluster's shape and spatial extent, they do not, in general, show the lines' geometries. To select a region in 3D space where lines are shown, the user can brush a region on a cluster hull, and this region is extruded to 3D in a specific way (c.f. Section 5). Via brushing the user increases a visibility parameter in the selected region, requesting additional lines besides the cluster representatives. If all lines passing through the selected region are shown concurrently, however, occlusions are quickly introduced and important structures are hidden (see Fig. 2 (c)).

To adapt the density of displayed lines in the selected focus region, we employ the CCF as a measure of information content (c.f. Section 5), as shown in Fig. 2 (d). Furthermore, since regions conveying high information can be missed when brushing manually, the visibility is transported automatically along the gradient in the CCF towards regions of low consistency (Fig. 2 (e)), or along bundles with low directional variance to reveal major trends in the line set. In either case, line density control is active to focus only on the most relevant lines.

4. Cluster Consistency Fields

CCFs are computed on the GPU using rasterization of lines into a Cartesian 3D voxel grid. Initially, we compute the resolution $r_x \times r_y \times r_z$ of the voxel grid into which the lines are voxelized. If the resolution is too high, a voxel carries merely information about one single line passing through it, yet if it is too low, the CCF cannot serve the purpose of a local indicator of consistency. In our implementation we link the average distance of the cluster representative to all lines in the cluster to the side length of cube-shaped voxels. Our experiments have shown that this choice gives a good balance between locality and information content. The grid size is set so that the aspect ratio of the bounding box of the lines per cluster is maintained. The vertex coordinates v_i are then transformed to local object coordinates in the range from $(0,0,0)$ to (r_x, r_y, r_z).

The voxel values can be optionally filtered using a low-pass filter with adjustable extent. In this way, high frequencies in whatever values are sampled can be removed, and a smooth consistency representation is obtained. In particular, if a feature is present in one voxel the smoothing process distributes this information into the surrounding region. In combination with our proposed brush-based focus visualization this leads to an improved continuation of features beyond the brush extent. For this reason we store a separate low-pass filtered version for every CCF we generate.

For each line in parallel, and starting with the first vertex, every pair of vertices v_i and v_{i+1} is processed consecutively. A line

through v_i and v_{i+1} is clipped against the voxel boundaries, via line-face intersection tests in the order of their occurrence from v_i to v_{i+1}. If v_i and v_{i+1} are located in the same voxel, no new intersection point is generated. This gives a sequence of voxel-face intersections, and every pair of consecutive intersections represents a line segment that enters into a voxel and exits that voxel. From all line segments that are generated for a single voxel, a number of different quantities are derived to generate different variants of CCFs. These variants are explained in the following.

4.1. Line density fields

For each line segment, the real length of the line from the entry to the exit point is computed and added into the CCF. Since in general the first and last vertex of a line do not lie on a face, we consider the length from the first vertex to the first face intersection and from the last face intersection to the last vertex in these cases.

Adding a length value to the CCF means to add this value to a counter at the voxel the line is currently passing through. Note that this is different to the construction of visitation volumes, where every segment adds the same contribution to a voxel regardless of its length. Our approach, in contrast, keeps track of the voxels' fill rates to obtain a more accurate measure of the line density per voxel.

The line density distribution within one cluster provides information about where and how many lines come close together (see Fig. 3 (left)). The level set to a threshold just above zero gives an approximate hull enclosing all lines. Regions in which only few isolated lines occur appear as low value regions which can be filtered out by slightly increasing the threshold. Then again, these regions can also be emphasized to show outliers which have been assigned to a cluster even though at least in some region they deviate strongly from the rest of the lines in that cluster. Fig. 3 (right) shows the cluster hulls that were rendered as level sets in multiple cluster density fields.

Level sets in the line density field are visualized using direct isosurface raycasting. For shading, gradients are calculated on-the-fly using central differences. The cluster hulls should be as transparent as possible without failing to communicate the outer shape. We therefore adapt the opacity of the isosurface depending on the angle between the surface normal and view direction [HGH*10]. In this way, the silhouettes of cluster hulls are enhanced while a non-obscured view into the cluster interior is provided otherwise.

4.2. Directional variance fields

Nearby lines in one single cluster that follow the same direction are in most cases less interesting than lines following different directions, they even carry redundant information and can thus be condensed. Xu et al. [XLS10] have motivated this statement in the context of vector fields via information entropy as a measure of how much information is carried by a region, depending on the directional uniformity of the vectors in this region. Measuring the variation of lines per volumetric unit allows to directly communicate locations which are potentially relevant for the understanding of the underlying line structure, at the same time determining regions where less lines than available can be shown. In complex line

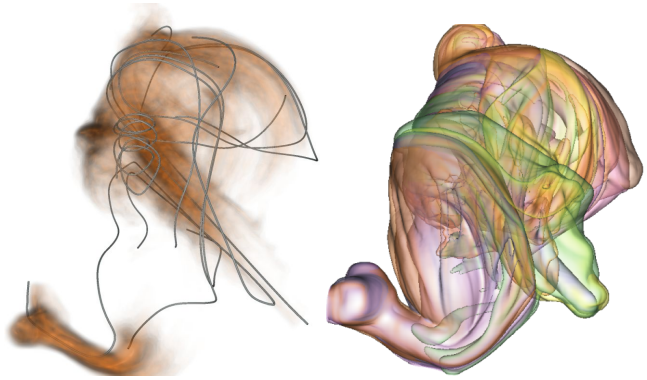

Figure 3: *Left: Representative lines of 8 clusters are shown in combination with summed up density of all clusters mapped linearly to opacity. Right: Hulls of 8 clusters using line density fields.*

sets, like turbulent structures, lines can deviate strongly in direction. By solely filtering line segments of low directional variance, the resulting image can still suffer from visual clutter. A measure of variation is therefore used as an additional filtering criterion for the visualization algorithm, and we use it in particular in combination with interactive brushing to balance the number of shown lines with respect to the information they carry. Furthermore, the measure is used to guide towards regions of high information content which have not been selected by the user. Similar in spirit to the approach by Xu et al. [XLS10], we calculate per voxel the directional variance of all line segments passing through that voxel. Let the line segments in a certain voxel be given as a set of direction vectors $\mathbf{d}_1,...,\mathbf{d}_n$. The directional variance, weighted by the length of each line segment, is defined as following:

$$\sigma(\mathbf{d}_1,...,\mathbf{d}_n) = 1 - ||\frac{\sum_i \mathbf{d}_i}{\sum_i ||\mathbf{d}_i||}|| \quad (1)$$

If the directional variance is high, σ is close to one, while it is zero if all directions are equal. This is demonstrated in Fig. 4, where the directional variance around the core of a tornado is shown. As can be seen, the variance is strongly increasing towards the core, with the magnitude of the gradient of the variance starting to grow strongly with decreasing distance from the core.

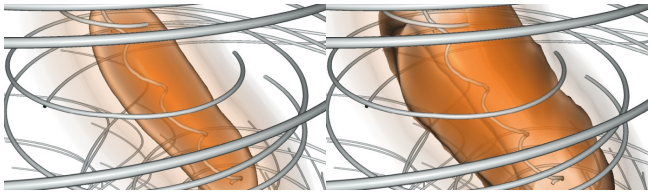

Figure 4: *Left: Cluster variance field of the Tornado data set. Color and opacity mapping is as for cluster density. Right: Magnitude of the gradient of the cluster variance field. The maximum gradients are located around the core of the tornado.*

4.3. Mean direction fields

The mean direction volume of a cluster stores at every voxel one representative average line segment; by averaging separately the start points and end points of all flow-aligned line segments in a voxel. This gives two average points which define the average line direction that is stored per voxel. The mean direction volume is not visualized directly but used for accessing the main flow direction per voxel in constant time.

5. Line Density Control via Brushing

We use the pre-computed CCFs to visualize cluster context information, i.e., cluster hulls, and to adapt the density of lines that are shown in a region that is brushed by the user. Cluster boundaries are visualized as isocontours in the cluster density field, and they support the user in identifying the boundaries of the brushable zone. At the same time they hint towards prominent regions, for instance, regions where the consistency of lines is extraordinarily low or high. This forms the context in which the user queries additional lines, and by this, analyses interactively the internal cluster structure. The user can request a more detailed representation of the line set by interactively using a brush in screen-space. Therefore, the 2D brush region is extruded to 3D along the viewing direction, defining a 3D brush volume in which the visibility is increased. Per default, the visibility is zero everywhere, meaning that only the cluster representatives are visualized. Where the visibility is increased, additional lines are shown, yet their density is controlled by the content of the CCFs as well as occlusion information that is computed for that location. The hierarchical decomposition of a selected cluster is utilized in combination with the visibility values to adapt the line density so that a sparse set of lines is shown where lines are close together, yet enough lines are shown to emphasize the important trends in regions where the cluster consistency is low.

Whenever the brush is moved a visibility volume is filled with visibility values $\in (0,1)$. Voxels in the brush volume get assigned a high visibility value. Simultaneously, via raycasting from every voxel center towards the viewplane in the cluster density field, the amount of occlusion, i.e., the accumulated density, a voxel receives due to lines in front of it is estimated. Voxels in the visibility volume that are not in the brush volume keep their values from previous frames, yet these values are continuously faded out over time if not queried once again. This allows one to move the camera while brushing without losing the previously brushed location, a mechanism that is demonstrated in Fig. 5, where the brush is applied and frozen before moving the camera to another viewpoint.

5.1. Local line density control

Based on the cluster tree every line gets assigned a *level-id*, which indicates the lowest level of the tree at which the line is chosen as representative. Note that for every line there is at least one level where this line is a representative line, at latest at the leaf node where the line is the only cluster member. The level-id is used during rendering to decide whether a line segment should be rendered in the selected focus region or not. Note that level-ids are divided by the depth of the cluster tree to obtain values in (0,1).

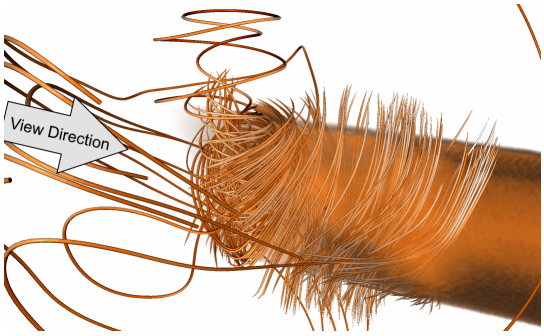

Figure 5: *The visibility volume is visualized using raycasting for demonstration purposes. The camera was originally positioned at the top left while using the brush. After brushing, the visibility values were frozen and the camera position was changed. With accumulated density samples, the visibility values are higher.*

3D line sets can be rendered efficiently via the rasterization-based rendering pipeline on the GPU, by constructing for every line segment a tube on-the-fly in a geometry shader resulting in 8×2 triangles per line segment. Every vertex that is generated when rendering a tube can access the visibility value at its object-space location, and this value can then be used to modify the rendering style of the tube segment or to discard the segment entirely. Every vertex compares the visibility value with the level-id of the line it belongs to. Only if the visibility value is larger than the level-id, the vertex is rendered, and it is discarded otherwise. In this way, the line density can be adapted automatically to the local object-space visibility, i.e., in regions of high visibility far more lines are rendered than in low visibility regions. In principle, also other rendering parameters can be controlled by in this way, for instance, opacity or color.

After all visibility values are resolved to per-vertex properties, the actual rendering is performed in two passes: In the first pass all lines are rasterized into a texture including their screen-space depth. We are using an intermediate vertex property for fading lines in or out to avoid sudden changes at the rendered image while moving the brush. Fading in or out is done by modifying the radius of the segment. The second pass performs raycasting and blends the previously rendered lines with the cluster hulls.

Visibility values for each voxel (x,y,z) in the brush volume (a cone with its center axis going through the camera position and the selected point in screen-space) are set by a compute shader according to

$$V_{x,y,z} = 1 - min(max(((\lambda + r) - p_d * D_{Acc} + p_v * Var), 0), 1) \quad (2)$$

With this formula, we provide a number of options to accentuate specific structures in the line set: λ defines a threshold that has to be exceeded before visibility values are set. By setting λ to a high value, more density values along the ray have to be accumulated or a region of high directional variance has to be passed, before visibility values are increased. Increasing λ enables a deeper insight into the data set. $r \in [0,1]$ is the distance of the ray to the center of the brush in screen-space. With this, threshold λ is increased with

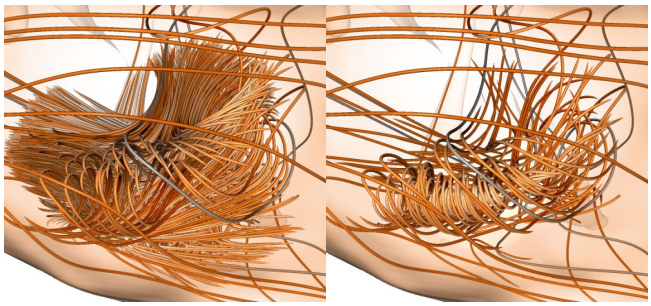

Figure 6: *By modifying parameters of the brush, different structures of the Aneurysm II data set are brought out. Left: Lines which are usually covered can be inspected. Right: Lines in regions of high variance are emphasized.*

increasing distance to the center of the brush, leading to a reduced density of lines at the border of the brush. D_{Acc} is the accumulated density along each ray from the camera to the corresponding voxel. The visibility values are increased with accumulated density, to not only show the nearest lines but to allow a placement of visibility values behind a number of lines as can be seen in Fig. 6 (left). The impact of this quantity is controlled by parameter p_d. Regions showing a high directional variance *Var* can be explicitly emphasized by increasing parameter p_v, as can be seen in Fig. 6 (right).

5.2. Brush guidance

By relying solely on the screen-space location of the used brush, the user can easily miss significant structures which are not in the brush volume, yet located very close to the brushed region in 3D space. In order to guide the user towards such structures, we propose a mechanism to automatically continue the selected brush volume anisotropically into their direction. This is achieved by letting visibility values being transported into the surrounding, either along gradient vectors in the CCF or lines passing through the selected cluster as illustrated in Fig. 7:

Mean Direction The visibility values are transported along the mean direction of lines, which is encoded in the mean direction field (see Fig. 7 (b)). By continuing the visibility values along the mean direction, crossing lines can be detected and the main path can be identified.

Variability Gradient By using the gradient field of the directional variance field, and transporting the visibility values along the gradient directions, nearby locations with high visibility will be revealed. This effect is shown in Fig. 7 (a), where the transport is towards the center of the vortex.

A mixture of both transport mechanisms is possible, by using them in combination but with different priorities. For instance, in the current implementation we use a linear blend between the gradient direction and the mean line direction according to the gradient magnitude. Altogether, visibility values are increased in the brush volume, transported into the dominating direction, and faded out over time. Thus, we can control the distance the visibility values can be continued via a global parameter that controls the life time of continuation.

The transport of visibility values is performed per frame in a compute shader. For each voxel, the visibility value and the vector indicating the location to which this value should be continued is computed. Instead of writing the visibility value to that location, we gather the visibility value from the location indicated by the inverse vector via tri-linear interpolation in the field of current visibility values.

6. Results

In the following, we show the quality of our method based on four data sets. All images were generated on a standard desktop computer (Intel 6x3.50 GHz processor, 32 GB RAM, and an NVIDIA GeForce GTX 1070 Ti graphics card). We have tested our method with following data sets:

Tornado 10000 streamlines were randomly seeded in a 256^3 vector field representing a tornado (Fig. 13).

Aneurysm I and II 4070 (Fig. 1 (right)) and 9200 streamlines (Fig. 11) describing the blood flow in two different aneurysms. [BMC14].

Turbulence 10000 streamlines were randomly seeded in a simulated turbulent vector field of size 1024^3 [AEE*16] (Fig. 12).

We compare our approach to importance-based line density control as proposed by Kanzler et al. [KFW16]. The lines of one cluster were extracted and set as input. The curvatures along the lines were mapped to importance values as proposed by the authors. The shading of lines was adjusted to focus on the distribution of line density. In Fig. 8 (left), we can see the overall density of lines adapted in a way to have an overview of the data set and to have an unobscured view towards the main vortex located at the left. With our brush-based approach in combination with refinement and transport of visibility values, the vortex at the left can be inspected (Fig. 8 (center)). By further inspecting the data set, another vortex at the bottom is revealed, which is not visible when using the importance criterion. The vortex is stretched resulting in low curvature and hence

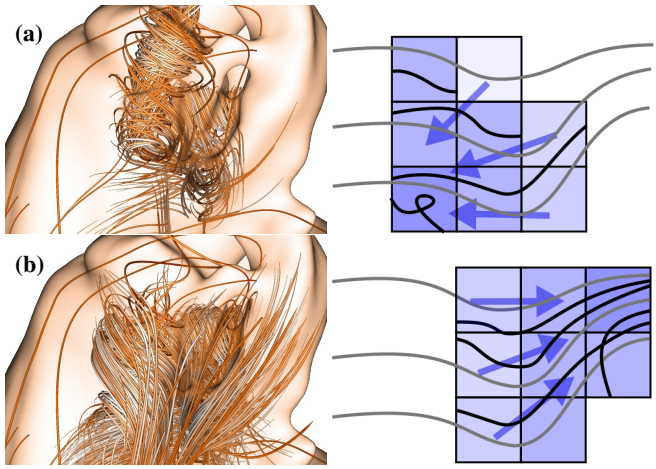

Figure 7: *Visibility values (blue) are controlling the line density. (a) Visibility values are transported to areas of higher variation. (b) Visibility values are transported along mean direction.*

Figure 8: *Comparison of importance-based density control as proposed by Kanzler et al. [KFW16] (left) and our brush based visualization (center and right). The vortex at the bottom of the image is not covered by Kanzler et al. but can be brought out using our brush (right).*

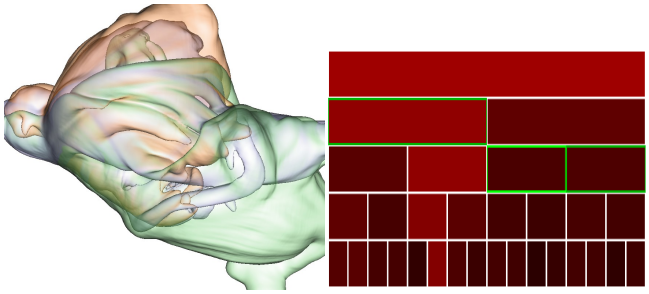

Figure 9: *On the left, 3 cluster hulls are visualized, which are selected in the cluster tree panel on the right.*

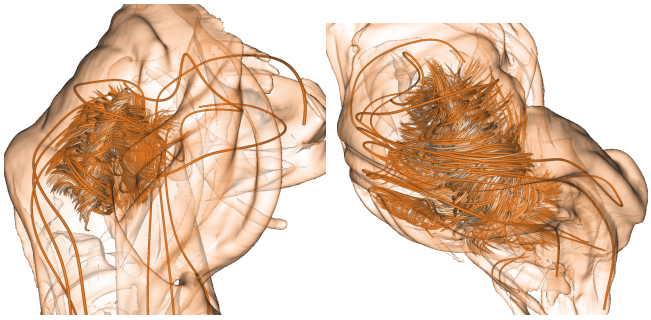

Figure 10: *The brush is used to add details by adding lines. Visibility values are placed by using primarily accumulated density and are extended in the direction of higher directional variance.*

a low importance value. Since our approach works in object-space, 3D flow structures are better preserved than the approach using a screen-space measure to adapt the line density.

The typical overall frame time (22 ms) includes line rendering (3 ms), rendering of cluster hulls using raycasting (13 ms), brushing at the focus region (4 ms), and visibility transport (2 ms) for a visibility volume of size 256^3. In all of our experiments the frame rate was consistently above 20 frames per second.

In the following, we describe a typical workflow using our method for the Aneurysm II data set. At first, the user selects a number of clusters for which the hulls are shown. This provides an overview of the clusters's extends and overlaps. Clusters can then be selected and further split into multiple clusters. A cluster panel represents the first levels of the cluster tree with its root at the top as shown in Fig. 9 on the right, and acts as control interface. The user is able to select and deselect different clusters at different levels, while the visual representation of the corresponding cluster hulls is updated immediately. The cluster panel communicates an estimate about the consistency of the cluster by using a mapping to a color gradient from black to red, i.e., from higher to lower consistency. The measure for consistency in this panel is calculated by summing up the distances of all lines of the cluster to the centroid of the cluster. By using this measure, a higher sum of distances indicates a cluster which might be a candidate for splitting up into two distinct clusters. After refining the clusters until a desired level, one can focus on one or two clusters by hiding the other clusters. To investigate one cluster in detail, the brush is applied as can be seen in Fig. 10. The vortex cores are now clearly visible while the hull in combination with auxiliary lines allows an estimate about the surroundings of the brushed location. By adapting the radius of the brush, the density of lines is increased in a larger region. By the automated continuation of the brush volume, additional structures become visible.

The Aneurysm II data set consists of long lines, resulting in cluster hulls that overlap in many regions. To further study at which location a cluster shares a feature with a nearby cluster, a second cluster can be enabled. The brush can now be applied to both clusters, revealing information about the underlying clustering. By transporting visibility values along the mean direction field and towards regions of high directional variance, cohesive structures can be identified: Fig. 11 shows the boundary between the vortex and the other cluster. In this example lines were not faded out to include the continuation of the lines forming the vortex.

Placing visibility values at regions of high directional variance by increasing parameter p_{var} and extending the selection along the lines, vortices embedded in turbulent structures can be revealed as can be seen in Fig. 12. By increasing λ, the coverage of lines located at the background is reduced. Combining this setting with a

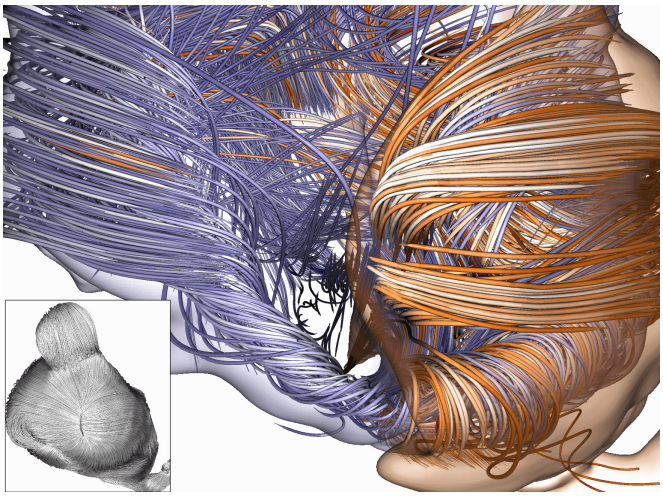

Figure 11: *The boundaries of two clusters are inspected by brushing. The vortex core at the bottom is split into two clusters. While the left part is consisting only of lines of the blue cluster, lines of the blue cluster are also interweaved in the orange cluster. Bottom left: All lines of the Aneurysm II data set.*

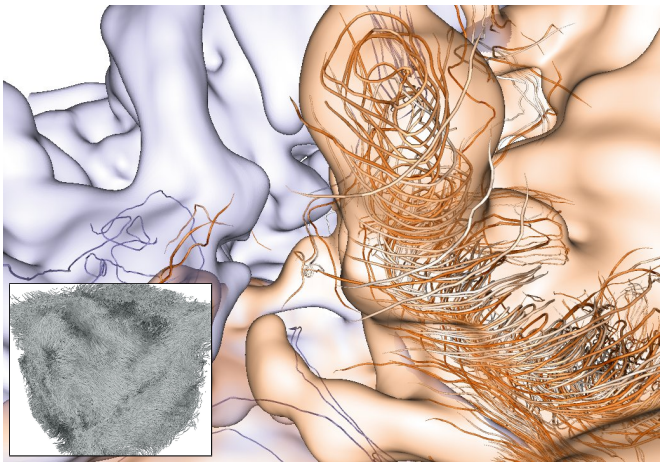

Figure 12: *Two clusters of the turbulence data set are inspected. A vortex structure is revealed by placing visibility values at regions of high directional variance. Bottom left: All lines of the Turbulence data set.*

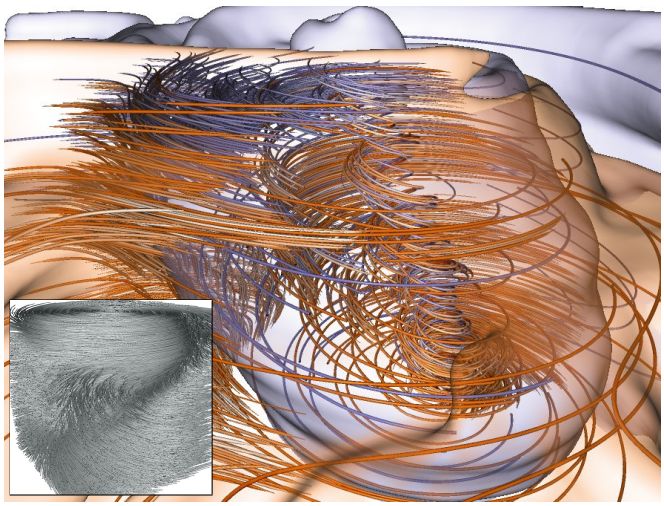

Figure 13: *Boundaries of two clusters are inspected by enhancing regions with high accumulated density inside the brush region. Transport of visibility values is showing the core of the tornado. Bottom left: All lines of the Tornado data set.*

visibility transport along the directional variance gradient, vortices in front of background lines are still emphasized as can be seen in Fig. 13 and Fig. 1 (right). The core vortex of the aneurysm is dominated by lines of one cluster while we can see a mixing of clusters at the core of the tornado.

7. Conclusion

In this paper, we have presented a novel approach to interactively explore clusters of lines via consistency-guided visualization techniques on the GPU. We have introduced cluster consistency fields and employed them to apply focus+context visualization techniques. We have shown the combination of cluster hulls with interactive and automatic techniques to select focus regions and adapt the line density to cluster consistency. This enables an in-depth comparison of cluster overlap regions and the line consistency of a selected cluster.

In future work, we will compare different clustering methods by matching similar clusters and simultaneously visualizing matches between two clustering methods. Specific characteristics of clustering methods can be studied directly on a given data set. We will further examine time-dependent data sets by finding a transition between brushed regions at different time steps. The development and movement of features is of interest in this case. Guiding a brush by exploiting time-dependent information can further enhance the understanding of the data set.

References

[AEE*16] ALUIE H., EYINK G., E. V., KANOV S. C. K., BURNS R., MENEVEAU C., SZALAY A.: Forced MHD turbulence data set. http://turbulence.pha.jhu.edu/docs/README-MHD.pdf, 2013 (accessed May 13, 2016). 6

[AMP10] ATEV S., MILLER G., PAPANIKOLOPOULOS N. P.: Clustering of vehicle trajectories. *IEEE Transactions on Intelligent Transportation Systems 11*, 3 (Sept 2010), 647–657. 1

[AT06] ANTONINI G., THIRAN J. P.: Counting pedestrians in video sequences using trajectory clustering. *IEEE Transactions on Circuits and Systems for Video Technology 16*, 8 (Aug 2006), 1008–1020. 1

[BBB*18] BEHRENDT B., BERG P., BEUING O., PREIM B., SAALFELD S.: Explorative blood flow visualization using dynamic line filtering based on surface features. 2

[BCM*08] BERMAN J. I., CHUNG S., MUKHERJEE P., HESS C. P., HAN E. T., HENRY R. G.: Probabilistic streamline q-ball tractography using the residual bootstrap. *NeuroImage 39*, 1 (2008), 215 – 222. 3

[BFMW12] BÜRGER K., FRAEDRICH R., MERHOF D., WESTERMANN R.: Instant visitation maps for interactive visualization of uncertain particle trajectories, 2012. 3

[BMC14] BYRNE G., MUT F., CEBRAL J.: Quantifying the large-scale hemodynamics of intracranial aneurysms. *Amer. J. of Neuroradiology 35* (2014), 333–338. 6

[CCK07] CHEN Y., COHEN J., KROLIK J.: Similarity-guided streamline placement with error evaluation. *IEEE Transactions on Visualization and Computer Graphics 13*, 6 (2007), 1448–1455. 2

[CGG04] COROUGE I., GOUTTARD S., GERIG G.: Towards a shape model of white matter fiber bundles using diffusion tensor MRI. In *IEEE International Symposium on Biomedical Imaging: Nano to Macro* (April 2004), pp. 344–347 Vol. 1. 3

[FG98] FUHRMANN A., GRÖLLER E.: Real-time techniques for 3d flow visualization. In *Proceedings of the conference on Visualization'98* (1998), IEEE Computer Society Press, pp. 305–312. 2

[GNBP11] GASTEIGER R., NEUGEBAUER M., BEUING O., PREIM B.: The flowlens: A focus-and-context visualization approach for exploration of blood flow in cerebral aneurysms. *IEEE Transactions on Visualization and Computer Graphics 17*, 12 (Dec. 2011), 2183–2192. 2

[GRT13] GÜNTHER T., RÖSSL C., THEISEL H.: Opacity optimization for 3D line fields. *Proc. ACM SIGGRAPH 32*, 4 (2013), 120. 2

[HGH*10] HUMMEL M., GARTH C., HAMANN B., HAGEN H., JOY K. I.: Iris: Illustrative rendering for integral surfaces. *IEEE Transactions on Visualization and Computer Graphics 16*, 6 (Nov 2010), 1319–1328. 4

[Jai10] JAIN A. K.: Data clustering: 50 years beyond k-means. *Pattern Recogn. Lett. 31*, 8 (2010), 651–666. 2

[JCK12] JACKSON B., COFFEY D., KEEFE D. F.: Force Brushes: Progressive Data-Driven Haptic Selection and Filtering for Multi-Variate Flow Visualizations. In *EuroVis - Short Papers* (2012), Meyer M., Weinkaufs T., (Eds.), The Eurographics Association. 2

[JL97] JOBARD B., LEFER W.: Creating evenly-spaced streamlines of arbitrary density. In *Proc. Eurographics Workshop on Visualization in Scientific Computing*. 1997, pp. 43–55. 2

[Jon08] JONES D. K.: Tractography gone wild: Probabilistic fibre tracking using the wild bootstrap with diffusion tensor mri. *IEEE Transactions on Medical Imaging 27*, 9 (Sept 2008), 1268–1274. 3

[KFW16] KANZLER M., FERSTL F., WESTERMANN R.: Line density control in screen-space via balanced line hierarchies. *Computers & Graphics 61* (2016), 29 – 39. 2, 3, 6, 7

[LHS08] LI L., HSIEH H.-H., SHEN H.-W.: Illustrative streamline placement and visualization. In *Proc. IEEE PacificVis* (2008), pp. 79–86. 2

[LMG06] LIU Z., MOORHEAD R., GRONER J.: An advanced evenly-spaced streamline placement algorithm. *IEEE Transactions on Visualization and Computer Graphics 12*, 5 (2006), 965–972. 2

[LMSC11] LEE T.-Y., MISHCHENKO O., SHEN H.-W., CRAWFIS R.: View point evaluation and streamline filtering for flow visualization. In *Proc. IEEE PacificVis* (2011), pp. 83–90. 2

[MCHM10] MARCHESIN S., CHEN C.-K., HO C., MA K.-L.: View-dependent streamlines for 3D vector fields. *IEEE Transactions on Visualization and Computer Graphics 16*, 6 (2010), 1578–1586. 2

[MHHI98] MAO X., HATANAKA Y., HIGASHIDA H., IMAMIYA A.: Image-guided streamline placement on curvilinear grid surfaces. In *Proc. IEEE Visualization* (1998), pp. 135–142. 2

[MJL*12] MCLOUGHLIN T., JONES M., LARAMEE R., MALKI R., MASTERS I., HANSEN C.: Similarity measures for enhancing interactive streamline seeding. *IEEE Transactions on Visualization and Computer Graphics 19*, 8 (2012), 1342–1353. 2

[MTHG03] MATTAUSCH O., THEUSSL T., HAUSER H., GRÖLLER E.: Strategies for interactive exploration of 3D flow using evenly-spaced illuminated streamlines. In *Proceedings of the 19th Spring Conference on Computer Graphics* (New York, NY, USA, 2003), ACM, pp. 213–222. 2

[MVvW05] MOBERTS B., VILANOVA A., VAN WIJK J.: Evaluation of fiber clustering methods for diffusion tensor imaging. In *Proc. IEEE Visualization* (Oct 2005), pp. 65–72. 1, 2

[MWS13] MA J., WANG C., SHENE C.-K.: Coherent view-dependent streamline selection for importance-driven flow visualization. *Proc. SPIE 8654, Visualization and Data Analysis* (2013). 2

[OLK*14] OELTZE S., LEHMANN D., KUHN A., JANIGA G., THEISEL H., PREIM B.: Blood flow clustering and applications in virtual stenting of intracranial aneurysms. *IEEE Transactions on Visualization and Computer Graphics 20*, 5 (2014), 686–701. 2

[RPP*09] ROSANWO O., PETZ C., PROHASKA S., HEGE H.-C., HOTZ I.: Dual streamline seeding. *Proc. IEEE PacificVis 0* (2009), 9–16. 2

[SHH*07] SCHLEMMER M., HOTZ I., HAMANN B., MORR F., HAGEN H.: Priority streamlines: A context-based visualization of flow fields. In *Proc. EG/IEEE VGTC EuroVis* (2007), pp. 227–234. 2

[SLCZ09] SPENCER B., LARAMEE R. S., CHEN G., ZHANG E.: Evenly spaced streamlines for surfaces: An image-based approach. *Computer Graphics Forum 28*, 6 (2009), 1618–1631. 2

[TB96] TURK G., BANKS D.: Image-guided streamline placement. In *Proc. ACM SIGGRAPH* (1996), pp. 453–460. 2

[VKP00] VERMA V., KAO D., PANG A.: A flow-guided streamline seeding strategy. In *Proc. IEEE Visualization* (2000), pp. 163–170. 2

[XLS10] XU L., LEE T.-Y., SHEN H.-W.: An information-theoretic framework for flow visualization. *IEEE Transactions on Visualization and Computer Graphics 16*, 6 (2010), 1216–1224. 2, 4

[YEII12] YU L., EFSTATHIOU K., ISENBERG P., ISENBERG T.: Efficient structure-aware selection techniques for 3D point cloud visualizations with 2DOF input. *IEEE Transactions on Visualization and Computer Graphics 18*, 12 (Dec. 2012), 2245–2254. 2

[YKP05] YE X., KAO D., PANG A.: Strategy for seeding 3D streamlines. In *Proc. IEEE Visualization 2005* (2005), pp. 471–478. 2

[YWSC12] YU H., WANG C., SHENE C.-K., CHEN J. H.: Hierarchical streamline bundles. *IEEE Transactions on Visualization and Computer Graphics 18*, 8 (2012), 1353–1367. 1, 2

[ZHT06] ZHANG Z., HUANG K., TAN T.: Comparison of similarity measures for trajectory clustering in outdoor surveillance scenes. In *Proc. International Conference on Pattern Recognition* (2006), vol. 3, pp. 1135–1138. 2